(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 279 652 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88301332.8

(22) Date of filing: 17.02.88

(51) Int. Cl.4: G 06 K 11/06

(30) Priority: 17.02.87 US 14943

(43) Date of publication of application:
24.08.88 Bulletin 88/34

(84) Designated Contracting States: DE FR GB

(71) Applicant: SENSOR FRAME INCORPORATED
Suite 3505 4516 Henry Street
Pittsburgh Pennsylvania 15213 (US)

(72) Inventor: McAvinney, Paul
2349 Tilbury Avenue Pittsburgh
Pennsylvania 15217 (US)

(74) Representative: Howden, Christopher Andrew et al
FORRESTER & BOEHMERT Widenmayerstrasse 4/I
D-8000 München 22 (DE)

(54) Method and apparatus for isolating and manipulating graphic objects on computer video monitor.

(57) A method of selection and/or manipulating graphic objects such as may appear on a computer video monitor is provided as is associated apparatus. A frame member cooperates with a plurality of optical sensors and a plurality of light occluding objects which tend to obstruct a portion of the light normally received by the optical sensors. The sensors are preferably dynamic RAM sensors. Determination of the position and orientation of the light occluding objects and, if desired, the velocity of the same, permits identification of or alteration of graphic objects. Fingers may be employed as the light occluding objects. This approach eliminates the need for burdensome identification of and changes to graphic objects through a computer keyboard or by means of a "mouse".

EP 0 279 652 A2

Description

"Method and apparatus for isolating and manipulating graphic objects on computer video monitor"

THE PRESENT INVENTION relates to a method and associated apparatus for detecting position, spatial orientation and/or velocity of a plurality of light occluding objects by optical means and, more specifically, is adapted to be used beneficially in respect of computer terminal display monitors.

It has been known to employ, in respect of computers, various devices in lieu of a conventional keyboard input for directing attention to a particular graphic object as by pointing. For example, light pens and the so-called "mouse" have been known as means for pointing to objects on a computer video monitor. It has also been known to permit such pointing to accomplish selection of alternatives from a limited menu.

It has been known to interrupt a plurality of beams with a single object and provide an indication of that single object. See, for example, U.S. Patents 3,673,327 and 4,517,559. It has also been known to interrupt a widely-dispersed light beam to block some subset of an array of sensors. See U.S. Patent 4,247,767.

It has also been known to employ a plurality of charge-coupled-devices (CCD) to determine the position within an x-y axis arrangement of a single object. See, also, "CCDs in Optical Touch Panels Deliver High Resolution", Electronic Design September 27, 1980, pp 139-143(a) and 43(b).

In spite of these prior art disclosures, there remains a very real and substantial need for a device which is adapted to provide a precision indication of position and spatial orientation of a plurality of objects and velocity of the same.

It is an object of the present invention to meet the above-described need and to provide a method and apparatus which will provide precise means for determining the position and spatial orientation and velocity of a plurality of graphic objects as hereinafter defined.

According to one aspect of the invention, there is provided a method of manipulating graphic objects comprising providing a frame, and a plurality of optical sensors generally facing the frame interior, illuminating at least a portion of the space defined by said frame, employing a plurality of light occluding objects to obstruct a portion of the light from reaching at least certain elements of said sensor, and emitting signals from said sensors corresponding to the positions of said light occluding objects with respect to said frame.

The method may employ providing a video monitor, an adjacent frame illuminating the space within the frame and a plurality of optical sensors generally facing the frame interior. The light occluding objects may be employed to isolate, select, refine or manipulate graphic objects on the video monitor.

According to another aspect of the invention, there is provided apparatus for manipulating graphic objects comprising frame means for at least partially receiving light occluding objects, light source means at least partially illuminating the region adjacent said frame means, sensor means operatively associated with said frame means for receiving at least a portion of said illumination and emitting responsive signals, and signal processing means for employing said sensor signals to determine the position of said light occluding objects.

The apparatus preferably has a frame disposed in front of a video monitor, illuminating means for providing uniform illumination and a plurality of dynamic RAM sensors for receiving at least a portion of the illumination.

The method and apparatus may be used for other purposes, such as producing music or another audible output, without using a video monitor.

Embodiments of the invention are described below by way of example with reference to the accompanying drawings, in which:-

FIGURE 1 is a schematic illustration of an apparatus embodying the present invention,

FIGURE 2 illustrates a form of sensor chip usable in embodiments of the present invention,

Figure 3 illustrates the sensor chip of Figure 2 with a light source focused on the chip.

Figure 4 respectively illustrates a hand partially occluding the light and the resultant pattern on the sensor chip.

Figure 5 is a schematic illustration showing the field of view of a first sensor.

Figure 6 is a schematic illustration showing the field of view of a second sensor.

Figure 7 is a schematic illustration showing the fields of view of both sensors of Figures 5 and 6.

Figure 8 is a schematic illustration showing the silhouette aspect of sensor measurement.

Figure 9 is a schematic illustration of the geometric relationships between a pair of sensors and the sensed object.

Figure 10 is a schematic illustration of a form of frame wherein two light occluding objects are present.

Figure 11 is a schematic illustration similar to Figure 10 but with the light occluding objects in a different position.

Figures 12a-12c are schematic illustrations of how fingers may be employed to select graphic objects.

Figure 13 is a schematic illustration showing how manipulation can be achieved through the use of fingers.

Figures 14a and 14b are schematic illustrations of multiple-finger control of instruments.

Figures 15a and 15b are schematic illustrations of the use of multiple-fingers in scaling.

Figure 16 is a cross-sectional, schematic illustration of a form of lens/optical sensor coupling construction.

Figure 17 is a partial schematic illustration of an embodiment of the present invention.

Figure 18 is a flow diagram of a user function employed to read the frame.

Figures 19 through 20 are flow diagrams of portions of software employable in the invention.

Figure 21 is a block diagram of a form of controller of the present invention.

Figure 22 is a flow diagram of a portion of the software employable in the invention.

Figures 23-39 are schematic illustrations showing the relationship between light occluding objects such as fingers and the light rays, and related notation systems.

Figures 40-43 are flow diagrams of portions of software employable in the invention. (Figures 41, 42a, 42b and 43 are to be combined by portions which have matching letters A through E.

As used herein, the term "graphic object(s)" means one or more textual elements, graphic elements or combinations of both displayed on a computer video monitor or other display screen or the like.

As used herein, the term "light occluding object(s)" means one or more opaque objects which occlude at least a portion of the light which would otherwise impinge on an optical sensor. The expression shall expressly include but not be limited to fingers, feet, elongated physical objects such as wands, pencils, brushes and other items suitable for such purposes.

As used herein the term "frame" or "frame member" means an assembly providing one or more sources of light and a plurality of sensors for receiving at least portions of such light regardless of frame size, including but not limited to continuous closed frames of rectangular, square, circular or other shapes, or frames having discontinuities, structures having no frame shape in the conventional sense but having the desired light source(s) and number of sensors in the desired positions.

Figure 1 shows a generally rectangular frame member 2 which may have a frame housing made of metal or a suitable resinous plastic, for example. It is preferably disposed in front of a video monitor of a computer in close adjacency thereto. The frame member 2 in the form illustrated, has a display image region 4 and may optionally have an offset region 6. The light source may advantageously be a neon tube which is excited by a light source exciter 10 and is positioned within a light source diffuser having sections 12, 14, 16, 18. If desired, a suitable light source filter may be provided in order to maintain the light within the desired intensity range. This serves to provide substantially uniform illumination for the entire inner opening of the frame.

In the form illustrated, a group of preferred optical sensors are provided in the corners of the frame and are mounted on printed circuit panels whose angular orientation is adjustable. Each optical sensor preferably has focussing lenses disposed within a suitable housing so as to focus precisely light received from the frame interior upon the sensor cells. Sensor assembly 20 has associated lens system 22, sensor assembly 24 has associated lens system 26, sensor assembly 28 has associated lens system 30 and sensor assembly 34 has associated lens system 36.

In order to facilitate maximum efficiency in use of the present system, it is preferred that the sensor assemblies 20, 24, 28, 34 make use of dynamic RAM chips. A 64K RAM, for example, may consist of two separate rectangular arrays of 32,768 cells each having 128 by 256 cells. It is preferred that a rectilinear, as distinguished from linear array, be employed in order to provide the z-axis positioning information desired in the present invention. By using more widely separated rows of the rectilinear array and providing a correspondingly thick light source, increased capability to measure the z-axis position and angle may be provided. The sensors function as cameras obtaining a silhouette picture. Each individual cell in this type of sensor will either have an output of zero if the light impinging on it is of sufficient predetermined intensity or will have an output of one if the intensity is not at such a level, A suitable solid state sensor for use in the present invention is that sold under the trade designation IS32 OpticRAM by Micron Technology, Inc. of Boise, Idaho.

The preferred sensors of the present invention use dynamic random-access memories as optical sensors. Among the advantages of these types of sensors over the solid-state optical sensors such as charge-coupled devices are that they allow uncomplicated random access to any point in the visual field in microseconds, they do not send gray-scale information, thereby facilitating ease of edge detection, they are easy to interface with and they are relatively inexpensive.

It is preferred that the preferred sensor employed in the present invention have rectilinear arrays of cells in order to facilitate determination of the position of objects in the x, y and z axes framework. In making the initial determination that an object is present in the screen region being monitored, only a linear array of the sensor elements needs to be employed. In making determinations regarding orientation such as the angle of a finger, for example, two or more linear arrays of sensor cells would be monitored. Use of the z-axis is then involved.

Referring still to Figure 1, the display image region 4 may be the area in which the graphic objects will be displayed from the computer. The offset region 6 which is not required, but is preferred, may advantageously be employed in those instances where it is desired that interpretations of the meaning of a gesture by a light occluding object or objects be fed back to the user in the form of some visible manifestation of the interpretation and that the visible manifestation be visible to the user without being occluded by the user's hand. The offset region 6 may be employed for this purpose. For example, if the graphic object is to be moved it might be displayed two inches to the right of the user's hand so that the hand will not block the user's view of the object. If the user now wants to move the object to the left edge of the screen, a two inch offset region at the left will allow the user to do this as the hand can be sensed beyond the left edge of the screen.

For convenience of reference herein, the combination of the frame 2, the controller, the light source, the diffuser and the sensors will be referred to by the trademark "Sensor Frame" or "Sensor Frame device".

Figure 2 shows a sensor chip of the type used in the present invention. The chip has two spaced arrays 37, 38 of sensor cells of equal size. Although the preferred sensor is a dynamic RAM chip, it may use any device which senses light using an array of discrete sensors, such as a charge-coupled device (CCD).

In a dynamic random access memory (DRAM) chip, a cell, which corresponds to a bit, may be in one of two states, conventionally referred to by those skilled in the art as a "one" or a "zero".

DRAM chips come in many sizes, most commonly containing a number of cells or bits, which is some power of two. The DRAM chips currently preferred for use in the Sensor Frame device are "64K DRAMS, which actually contain 65,536 bits, arranged as two rectangles (see Figure 2), each containing 32,768 bits, or cells. The cells are sensitive to light in the "near infrared" region of the spectrum. Each 32K-bit rectangle is arranged as 128 rows and 256 columns.

Larger (or smaller) DRAMS could easily be used, if desired. Each DRAM has one input line (i.e. wire) for reading data (a 1 or 0) into a cell; and one output line for reading data out of the cell. A cell is selected for reading or writing by presenting an "address" (which in the case of a 64K DRAM is a 16-bit number) to the DRAM. While in practice, most 64K DRAMS have only 8 "address lines" for reading an address, the method of presenting a 16-bit address to a DRAM with only 8 address lines is called "multiplexing" and is well known to those skilled in the art.

Once an address is presented to a DRAM, a 1 or 0 may be written into it, or a previously stored value (either a 1 or 0) may be read out.

In the Sensor Frame device, "ones" are written into those DRAM cells which correspond to locations in the visual field to be scanned. The DRAM cells are then exposed to light, with the exception of those that are blocked by a finger or other occluding object. The individual cells consist partially of very small capacitors, which hold an electrical charge corresponding to a "1". All cells with "ones" written into them will eventually lose their charge, but those exposed to near infrared light will lose charge much more quickly, becoming "zeroes". With a light source of appropriate intensity, and a carefully-selected exposure time, cells which are blocked by fingers will contain "ones", and cells not blocked will contain "zeroes" after exposure.

In the present invention a fixed, continuous light source is used to provide a pattern of light (usually, but not necessarily, a rectangle), which is focussed on the chip. Figure 3 shows the light from a uniform-intensity rectangular light source focused by a lens system (not shown) on one of the sensor chips used in a sensor frame. The white bar 39 represents that portion of the chip upon which the light from the rectangular light source falls.

If one or more objects, such as human fingers 40, are placed between the light source 41 and the lens/sensor assembly 42, the objects will occlude light from part of the light source. The shadow of the objects provides, in effect, a silhouette, or outline of the shape of the light-occluding objects on the sensors. Figure 4 illustrates this phenomenon.

Although it might seem preferable to eliminate the need for a light source by forming an image using light reflected from the object, as in a camera, this method is not preferred for several reasons. First, the intensity of reflected light depends upon two factors which are extremely difficult to control in practice i.e. the reflectivity of the sensed object and the intensity of ambient light. Furthermore, even when very high-intensity light is used for object illumination, the intensity of reflected light, and the image contrast available, will be considerably less than that obtainable using the light-occlusion method described herein. This state of affairs tends to lengthen the required exposure time, making anything like real-time performance and image disambiguation extremely difficult outside a laboratory environment.

Because of optical design constraints, it is generally not practical to sense a ninety-degree field of view from each sensor, nor is it practical to use the information from a sensor which is very close to the sensed object. For this reason when calculating position, the present invention uses information only from those sensors which reside on the edge of the frame generally opposite the object. For example, if the aspect ratio of a rectangular Sensor Frame device is such that it is wider than it is high, the top and bottom left sensors will be used to calculate position of objects in the right half of the frame, and the top and bottom right sensors will be used to calculate position of objects in the left half of the frame. Figures 5 through 7 illustrate this case, with the cross-hatched areas within the frame showing the areas scanned by each sensor 43, 44.

Position of the occluding objects is determined by pairing adjacent sensors (usually, but not always, the two adjacent sensors on a short side of a rectangular frame), and then using the known distance between the sensors (the baseline distance) and the interior angles formed by lines passing from each sensor through the visual center of the object. This "angle-side-angle" information is sufficient to permit calculation of the x,y position of an object.

Figures 8 and 9 show this process in more detail. Figure 8 shows that the sensor 43 actually sees only the left and right edges 44, 45 of an occluding object 46. The Sensor Frame device controller stores numbers which represent the angles between the baseline 47 (the side of the frame between two sensors which are paired for purposes of position determination) and the left and right edges of the sensed object 46. Two numbers are stored; the first number corresponds to the angle between the baseline and the left edge of the object, and the other number corresponds to the angle between the baseline and the right edge of the object. This pair of edges is stored by the controller for each object seen by each sensor. The controller's internal table containing these numbers is called the raylist. Each

left-edge-angle/right-edge-angle pair is an entry in the raylist and is called a "ray".

The average of the left-edge angle and the right-edge angle for an object represents the angle of a line passing through the visual center 48 of that object (Figs. 8 and 9). The visual-center angles to an object from two adjacent sensors are used to calculate an object's position.

Figure 9 shows angle A, the angle between the baseline 47 and the line 49 passing from the top-left sensor through the visual center of the object as seen from the top-left sensor; and angle B, the angle between the baseline and the line 50 passing from the bottom left sensor through the visual center 48 of the object as seen from the bottom-left sensor. Angles A and B and the known length of the baseline 47 are "plugged into" the angle-side-angle trigonometric formulas to obtain readily the x and y position of the visual center of the object.

One of the objectives of the present invention is to detect positions, angles, and velocities of multiple light occluding objects in real-time. It is desired, for example, to provide an inexpensive means of recognizing rich classes of complex gestures quickly enough to control a vehicle or musical instrument in real-time.

There are two fundamentally different approaches to optical, or vision-based gesture recognition. These may be classified as the "perspective scan" and "peripheral scan" methods. The perspective scan method places a video camera either above or below the user's hand. The advantage of this approach is that the whole hand can be seen at once. The disadvantage is that it is slower and about an order of magnitude more expensive, in terms of hardware costs, than the approach described herein.

The Sensor Frame device is a peripheral-scan sensing device. Although it is inherently much faster than a perspective-scan device, it presents two potential problems. One potential problem is occlusion. In a peripheral-scan device, the user's fingers may block one-another, making tracking difficult. The architecture of the Sensor Frame device makes it possible to track occluded (blocked) fingers by maintaining a history of the movement of their edges. A second potential problem is ambiguity. It is possible for two different finger position configurations to produce the same pattern on a pair of sensors. This aliasing effect will be referred to herein as "pattern ambiguity". The problem can be solved using a method which will be referred to as "temporal disambiguation". Briefly stated, one can select between two or more possible finger configurations by scanning quickly enough i.e. - with high-enough temporal resolution, to know which fingers arrived first in the visual field, and therefore which fingers are real.

Figure 10 shows the effect of two fingers 81, 82 entering the visual field of the pair of sensors 28, 34 on the right. Between the fingers and the sensors are the "rays" produced by the silhouettes of the fingers occluding the light source. For each scan of the Sensor Frame device visual field, a list of these rays (called a raylist) is compiled for each sensor. Note that there are four areas A, B, C, D where the rays overlap. The same ray overlap pattern can be created by either a vertical arrangement of fingers, as shown (the fingers are at positions 81 and 82 in Figure 10), or a horizontal arrangement (as with fingers 83 and 84 in Figure 11), resulting in ambiguity regarding which of the two patterns is the real one. This pattern ambiguity may be resolved by a number of means. These include but are not limited to: temporal disambiguation i.e. -keeping track of which finger arrived first, use of additional sensors, and knowledge-based disocclusion, wherein knowledge of user physiology and/or tools, such as the size of the user's fingers, is used.

If it were desired to employ the system to rotate a graphic object 90 degrees, for example, a user could position two fingers in the vicinity of the video monitor which contained the graphic object. The fingers could then be rotated as a unit. The system would determine the initial location, the direction and the extent of rotation and would, through the computer, effect responsive rotation of the graphic object in the desired direction through a 90 degree arc.

A further example of uses of the system would be to have fingers point to a graphic object on a monitor screen and determine the angle of the fingers with respect to the z-axis. This could serve as an instruction to the computer to move that particular graphic object, and the finger angle relative to the z axis would determine the direction in which it is to be moved. Withdrawal of the finger could serve to provide an instruction to stop moving the graphic object.

Figures 12a through 12c show how multiple (in this case two) fingers might be used to select the displayed objects upon which to perform an operation. In Figure 12a frame at left are a square 50, a triangle 51, and a circle 52. In Figure 12b, the user's fingers 53,54 are used to determine the diagonally-opposite corners of a rectangle 55, whose size and position are determined by the fingers 53, 54. Any object falling completely inside the rectangle 55 will be operated upon, and any object not completely within the rectangle 55 will not be operated upon. In Figure 12b, for example, the square 50 and triangle 51 are selected, but the circle 62 is not. In Figure 12c, all three objects are within the rectangle 56 and all are selected.

Figure 13 shows how two fingers can be used to rotate a knob in a control panel displayed on a video monitor. Row (a) shows a knob 56 with five detent positions, pointing respectively to positions 3, 4 and 5. In row (b), as fingers 57, 58 are rotated in a clockwise fashion, the Sensor Frame device provides their positions to the panel-control program. The knob 56 turns responsively to reflect the position of the fingers 57, 58. Although the row (b) sequence shows the fingers 57, 58 positioned above and below the knob, exact positioning need not be critical; and in sequence, the same effect can be achieved if one finger 57 is over the center of the knob 56 and the other finger 58 is at the knob periphery, for example (See row (c)). All that is really important in this application of the Sensor Frame

device is that the fingers be in the vicinity of the knob, and that they be rotated through an angle sufficient to cause the knob to rotate to the desired detent position.

Figures 14a and 14b illustrate how multiple-finger gestures can be used to control sophisticated instruments such as oscilliscopes, logic analyzers, or waveform function generators. In these figures, two fingers 59, 60 are used to control the frequency and amplitude of a square wave 61 produced by a waveform function generator.

In Figure 14a, the user has placed fingers 59, 60 respectively near the top and bottom peaks of the displayed waveform. Figure 14b shows what happens when the fingers 59, 60 are moved apart. The amplitude of the generated waveform changes (in this case it increases) in proportion to the movement of the fingers 59, 60 to create waveform 61'.

Figures 15a, 15b show the same method applied to frequency scaling. The user has oriented two fingers 62, 63 horizontally in the left illustration, and then moved them closer to each other, decreasing the period (and therefore increasing the frequency) of the generated waveform 64, 64'.

It will be appreciated that these figures show examples of the "control panel" applications of the Sensor Frame device and that other uses and more sophisticated uses of these types of gestures can be achieved.

Referring to Figure 16 there is shown a form of optical system and dynamic RAM sensor arrangement preferred for use as sensors in the present invention. Parallel rays from the frame interior are shown entering as indicated by the arrow at "L". A pair of spaced focussing lenses 52, 54 causes the entering light to focus at 56 so as to focus an image of occluding objects disposed in the frame onto the sensor array 60. By way of example, a primary lens at the front may be a 3/8" focal-length double concave lens and the secondary lens 54 a 1/4" focal-length double convex lens.

Figure 17 illustrates a "stand-alone" embodiment of the invention. A host computer 60 is connected to controller 64 by a cable 66. The DC supply and light-source synchronization component 70 is connected to the controller 64 by a cable 71. High voltage AC power supply 73 is connected to the frame light source by a high voltage cable 74. Controller 64 is connected to the frame sensor assemblies 20, 24, 28, 34 by cable 78. In operation, the high-voltage supply 73 energizes the light source. If desired, the controller 64 may be provided as a component within the housing of the host computer 60.

When an object such as a human finger blocks the image of the light source, a shadow with edges corresponding to the edges of the occluded object appears in the sensor's focused image. The term "ray" as used herein refers to the shadow cast by the occluding object. As the object moves, the rays move correspondingly.

In one embodiment of the invention the host computer 60 has the (a) "TrackObjects" function which determines the x, y and z axis coordinates of the occluding objects and (b) the "TrackRays" function which maintains a history of the rays for each sensor. The preferred embodiment, however, executes these functions in the controller 64.

The controller 64 builds raylists which are lists of rays seen in the latest exposure of the sensors. The raylists may be transferred from the controller 64 to the host computer 60 as input to the ray tracking and object tracking software. This transfer may be effected about every 30-35 milliseconds, for example. The raylist is established in the controller 64 by scanning the sensors. The host computer 60 maintains the raylists for each sensor. "Raypaths" contain the last n samples extracted from the raylist for a given sensor for a given object, where n is application dependent. A linked list of raypaths for each sensor is referred to as a path list. A "simple raypath" is one sensor's completely unoccluded view of an object, such as a finger, for example. A "cluster" is a raypath having a doubly-linked sublist of raypaths wherein each entry in the sublist is a raypath representing one sensor's view of a partially occluded or fully occluded object. By comparing the information provided in each subsequent raylist with prior raylists the changes in object position, orientation and velocity can be determined. This information can be employed to alter a graphic object on the video monitor.

It will be appreciated that various means for providing software to perform the functions required in the method and apparatus of the present invention may be employed. The following will illustrate and describe what is presently deemed by applicant to be the best mode of accomplishing the same.

Figure 18 shows a flow diagram for an application function employed to obtain a list of objects from the Sensor Frame device. At the beginning of the ReadSensorFrame function 80 it is determined at 82 whether the Sensor Frame has any new data. If so, an updated raylist is obtained from controller 64 by calling the GetRayList function 84 and then the TrackObjects function 86 is called. If any objects are found in the list returned by the TrackObjects function 88, the user program displays the objects on the video terminal using the DisplayObjects function 90. If no objects are found, the function returns 92.

Figure 19 illustrates a flow diagram for the TrackObjects function for use in tracking objects. This function employs four other functions to identify new objects and track the movements of the objects already known.

At the beginning of the TrackObjects function 100 the "ray indices" are incremented 102. This causes all updated ray information to be written into a circular buffer, allowing information about previous samples of a ray to be retained until they are no longer needed. Past ray information is used, for example, to obtain velocity by taking the difference between a current ray position and a previous position.

The FilterRays function 104 eliminates rays that are too small or too large to be of interest. For example, if a human finger is known to be an object, greater tracking speed and reliability can be achieved if one eliminates consideration of rays

which are clearly too narrow to represent the human finger. The filter parameters are generally application dependent and may be modified by user programs.

The TrackRays function 106 tracks the motion of rays which are related to previously known objects and creates a new raypath when a ray is found in the raylist which does not match a ray found in the past. This serves to update positions of rays, check overlap of old and new rays, and recognize merging and splitting of rays such as would occur when one object passes behind another.

The BindDuals function 108, for each pair of sensors tries to find a new raypath for each sensor of the pair with the constraint that the two raypaths must have been created closely together in time. When the BindDuals function 108 finds a pair of raypaths which were created with a sufficiently short time period, it assumes that they are caused by the same object and creates an object data structure to represent the new object. The two raypaths are then bound to the new object which survives until the raypaths bound to it have been deleted. At this point 110, if there are no objects, the TrackObjects function returns control to the calling function 114.

The UpdateExistingObjects function 112 plays several roles. Where clustered fingers are detected, it checks to see if the size of an object is large enough such that more than one finger should be assumed to be present. It also plots the current position of objects based on the latest information in the raypath structures which have previously been updated by the TrackRays function 104. It also looks for raypaths that have not been previously bound to an object and attempts to bind them to an existing object. Also, the UpdateExistingObjects function 112, when it detects an object structure from each pair of sensors each representing the same physical object, attempts to combine the two object structures into one. It succeeds only if the absolute x-y plane positions of both objects are sufficiently close together.

Referring to Figure 20 there is shown a flow diagram which illustrates a preferred manner of having the controller 64 (Fig. 17) scan the visual field produced by the sensors to obtain raylists.

In Figure 21, the Snapshot function 130 waits 132 until the light source (which may be a neon or fluorescent tube which preferably flashes on and off at a rate of 60Hz) is at a known brightness level. This is accomplished by synchronizing the sensor write/scan cycle to the 60Hz line frequency used to power the light source. An interrupt from the synchronization circuit 150, 151 (Fig. 18) indicates readiness for an exposure.

After writing "ones" 134 to the appropriate sensor cells to establish them at the "1" level, the Snapshot routine sends a signal 136 to the AnalyzeSema function 138 that it may start processing the results of the previous exposure. While the Snapshot function 130 is waiting for the sensor cells to be exposed 146, the AnalyzeSema function 138 can process the results of the previous exposure 140 and call the SendRays function 142 which send raylist to the host computer 60 by way of the host interface controller 152 of Figure 21. In this fashion, exposure takes place in parallel with low-level image processing at the controller level. Upon completion of the exposure of the sensors, Snapshot is rescheduled 146 as a consequence of a timer interrupt to the control microprocessor 153 in Figure 21 from the clock/counter 151 in Figure 21. It then resets image interrupts 148, and calls the scanner function 150 to read the sensor cells into which 1's were earlier written. Cells which were exposed to light from the light source will be at zero and those blocked by the objects to be sensed will have retained their charge and will, therefore, be read as one's.

The scanner function 150 achieves high efficiency and, therefore, high exposure rates using a combination of software and hardware. The reading of the sensor cells is preferably done not by a loop but by an unfolded sequence of read instructions, one for each cell. Each read instruction reads the same cell for all configured sensors, in parallel. There are no instructions to test the state of the data read in by the read instructions. Eliminating the test and conditional branch instructions makes the scan about three times faster than it would otherwise be.

Referring still to Figure 21, before the scanner function 150 reads data from the sensors, it writes all zeroes to the lowest-order N bits of the control register 156, where N is the number of sensors employed in the sensor frame device. When a read of the sensors is subsequently done, the incoming data from the sensors, appearing in the N-bit data buffer 157 is compared by the comparator 155 with the data previously stored in the control register 156. If the newly-read data 157 does not match the previous data 156, the image interrupt latch 154 is set, causing an interrupt to the controller microprocessor 153. The microprocessor 153 cooperates with the clock/counter 151, host interface controller 152, ROM/RAM 148 and light source synch circuit 158.

When this occurs, an image interrupt handling routine uses the read-instruction address at which the interrupt occurred to calculate the column address of the sensor cell at which the pattern change occurred. The newly read N-bit pattern is compared with the pattern previously stored in the control register 156 and for each bit which is different, the column number of the sensor read is stored as an edge in the raylist for the sensor whose input was changed. The new N-bit pattern is then stored in the lowest N bits of the control register 156, and the interrupt handler returns control to the scanner routine which continues reading until another pattern change causes an interrupt, or until all cells in a column of the sensor 150 have been read.

Figure 22 illustrates a flow diagram for the TrackRays function. At the beginning of TrackRays 160, the sensor number is set to 0, and the first raypath is fetched for this sensor 164. If there are no raypaths 166, the sensor number is incremented. If all sensors have been checked for non-empty path lists 170, the FindOrphans function 172 is called to find all rays which could not be matched with already-existing rays (this is how new rays are discovered). After FindOrphans is called, the TrackRays function returns 174. For each sensor having a

non-empty list of raypaths, the function TrackRays traverses the raypath list twice. In the first pass (178 to 182) the MatchRay function 178 is called. It takes the previously recorded ray position and uses previously recorded ray velocity information to estimate the ray's current position. If there is no match, the ray is marked for release 190. If the path has remained unmatched for several scans, it is released, and if it is the last path attached to its object, the object is released 190. If there are no more paths in the path list 182 and if there are any unmatched paths 184, an attempt is made to match the rays assuming a higher velocity 192. If on the first traversal of the raypath list any clusters are encountered, a second pass through the list is made (196 through 204). Raypaths which are identified as clusters contain a doubly-linked sublist of raypaths representing a cluster of occluded objects as seen from one sensor. The UpdateCluster function 200 is called to traverse this sublist, calculating the positions of the rays within the cluster based upon both the left and right edges of the current cluster position and on the edge and object binding attributes of the raypath within the cluster. If the size of a cluster is found to have gone through a local minimum, the CheckForReversal function 202 may be employed to reverse the left-to-right edge-binding attributes of each raypath within the cluster. Edge and object binding attributes will be discussed hereinafter.

The MatchRay function 178 also checks to see if one of two special conditions have occurred. First, if the current raypath sample overlaps more than one ray in the new raylist, a split of the ray cluster has occurred. The raypaths cluster is split into two parts. It splits a raypath cluster into two parts, the position of the split depending upon the position of the newly discovered gap in the raypath cluster currently being matched. Secondly, if a ray in the new raylist overlaps more than one raypath of a given sensor, a merge of two objects (or clusters of objects) has occurred. The two adjacent raypaths are emerged into the same cluster, care being taken to record the raypath "edge-binding" and "object-binding" attributes.

The sensors of the Sensor Frame device focus an image of the frame's light source onto the sensor chip. Where an object such as a human finger blocks the image of the light source, a shadow appears in the focused image. The edges of the shadow correspond to the edges of the occluding object. The term "ray", as used herein, refers to the shadow cast by an occluding object. As the object moves, its ray moves correspondingly.

Figure 23 is a schematic representation of the angles swept out by the two rays produced by two objects 207, 208 in the visual field of the Sensor Frame device. The view is from a position looking into the video display terminal. Figure 23 shows the rays sensed by a sensor. Figure 24 shows the rays sensed by two sensors, the other sensor being located in the lower right-hand corner. Because the rays cross in four places rather than two, it is seen that the same pattern of rays can be produced by more than one arrangement of fingers as in, for example, Figure 25.

In Figure 26, there is shown a dotted line 209 to show the edge of the field of view of the bottom-left sensor. Each pair of sensors along each short edge of the Sensor Frame device looks at objects in the opposite half of the image area. Therefore in the case shown, the two bottom sensors look at the top half of the image area, and the two top sensors look at the bottom half of the image area.

For convenience of reference, diagrams of the type shown in Figures 23 through 27 will subsequently be referred to as overview notation for the representation of sensed objects.

Although overview notation is employed in order to represent objects as seen by more than one sensor, there is a simpler notation for representing the image of one or more objects as seen by only one sensor. This concise representation, which we will call ray-transition notation or RTN, is shown in Figure 28. RTN makes it easy to represent sequences of finger positions, just as frames in a strip of movie film are used to give the illusion of motion. In RTN, each frame shows the position of the objects seen by one sensor at a given time. Figure 28 shows two adjacent frames, or "snapshots" of fingers, at two instants of time. The fingers are represented by the bracket-like objects. One can think of the sensor as sitting off to the left side of the frame, and in the plane of the frame. In this case, an object shown nearer the top of the frame is closer to the left side of the image seen by a sensor, and an object nearer the bottom of the frame is closer to the right side of the sensor image.

The RTN images taken at time 1 and time 2 in Figure 28 correspond to the positions shown in Figures 26 and 27, respectively, as 207', 208' and 207" and 208".

Figures 29 and 30 show an occluded object (object 2) in both notations. Again, in the case of the RTN notation, if we think of the sensor as sitting to the left of the RTN frame in the plane of the frame, object 2 will be "behind" object 3.

It is important to remember that the Sensor Frame device, in the preferred embodiment, does not produce gray-scale images. Everything is black or white. This means that if two objects move together, as shown in Figure 31, their interior edges disappear, leaving the impression of one larger object. Of course, it is desired to maintain the distinction between the two objects in spite of this edge occlusion, and the RTN representation of Figure 32 at "Time 2" continues to show the edge between objects 1 and 2; similarly, knowledge of occluded edges is maintained by the Sensor Frame tracking algorithm. Figures 33 and 34 show the reverse of the merging of two objects, known as a split.

With these notations available for describing the spatial relations of objects as seen by the Sensor Frame device, one is now in a position to describe how ray and object tracking are accomplished in the present invention.

In the Sensor Frame device, the layer of the software which returns the x and y-axis position, and angle of an object about the z axis, is the "object-tracking" layer, embodied in the function

TrackObjects.

Below the object tracking layer is the ray tracking layer, embodied in the function TrackRays which maintains a history of rays for each sensor, and deals with merging the rays when one object moves behind another, and splitting of rays when one object moves out from behind another.

There are data structures common to both the object-tracking levels. They are: (1) RayLists - For each sensor we maintain a list of rays (and their edges) see in the latest exposure of the sensors. This is the lowest level "raw" data produced by a sensor scan. (2) Raypaths and Path Lists - For each sensor we maintain a linked list of nodes called raypaths. Each raypath contains the last n raylist samples extracted from the raylist for a given object, where n is application-dependent. The linked list of raypaths kept for each sensor is called a path list. An entry in the path list may be a simple raypath or a cluster. A simple raypath represents one sensor's completely unoccluded view of an object (such as a finger). A cluster is a raypath having a doubly-linked sublist of raypaths. Each entry in the sublist is a raypath representing one sensor's view of a partially or fully-occluded object. (3) Objects - Object structures correspond to each physical object discovered by the object-tracking algorithm. For each sensor in whose visual field an object lies (whether or not it is occluded), the internal object structure contains a non-null pointer to a raypath.

A few examples will be given to clarify the purpose and use of these data structures.

Figure 35 shows four sequential "snapshots" of an occluded finger. In this case, finger 2 passes behind finger 1, as seen from the left sensor. Below each RTN frame representing the sensed objects is a structural representation of the lists of raypaths used to keep track of object motion at the sensor level.

When a "merge" of the two fingers occurs in frame 2, the current raypath list for the merged rays is pushed down one level, and a new node, representing the clustered raypaths, is inserted at the top level. When finger 2 emerges from behind finger 1 in frame 4, the single ray corresponding to the top-level cluster node splits into two again; the cluster node added in frame 2 is now discarded, and the two original raypath nodes emerge at the top level. If, while the two fingers had been merged, another ray had become occluded, its raypath would have been linked into the list of occluded raypaths with the two preexisting raypaths.

In addition, fingers or other objects will not always make their first appearance unoccluded, as in the above example. In many cases, the existence of multiple fingers in a cluster must be "inferred" if the application demands it.

At the top of Figure 36 is shown an overview-notation diagram of three fingers. Although the left sensor (sensor 1) sees all three fingers distinctly, fingers 1 and 2 are clustered together into one ray from the point of view of the right sensor (sensor 0).

As each sensor has only about a 60-degree field of view, dotted lines have been used in the overview diagram to delineate areas of the visual field in terms of the number of sensors that can sense objects in each area. For example, the "4S" designation in the diamond-shaped area at the center of the overview drawing indicates that four sensors can see an object in that area. Similarly, the "2S" designation in the area populated by the three objects in our example indicate that only two sensors, the bottom pair, can see the three objects.

Figure 38 is an example of the manner in which raylists, raypaths, and path lists, and objects are related within the internal Sensor Frame device data structure.

The representation of an object, in this case the object representing finger 2, is shown in the upper left corner of Figure 38. The object structure contains, among other things, a pointer to a raypath containing the history of the ray which matches finger 2. This is done for each sensor that can "see" finger 2. Therefore, the object contains null pointers for sensors 2 and 3 (at the top of the frame) because finger 2 cannot be seen by sensors 2 and 3.

In this internal-data-structure diagram, each small rectangular box with a ray number in it represents one of the "raw rays" sensed by the Sensor Frame's optical sensors. A collection of raw rays is called a raylist. In the example, the raylist contains three raw rays for the left sensor, and two for the right sensor. The raylists maintained for each sensor are the lowest level of representation in the sensor frame data structure. Each "snapshot" of the visual field produces a new raylist for each sensor. Each ray in a raylist is used to update the state of one or more raypaths, represented by circles in the data-structure diagram.

When a new raylist becomes available (about every 33 milliseconds, for example), the raypaths attached to existing objects are compared with each new ray to see if there is overlap. In the data-structure diagram, the curved arrows from raw ray 0 of the right sensor raylist to raypaths 1 and 2 indicate that both raypaths match the raw ray, and a case of occlusion exists, or "clustering" of multiple objects within one ray. The curved arrows only indicate a matching. They do not imply that there are pointers from the raw raylist to the raypaths.

The raypath clusters have attributes which allow us to make inferences about the position of a ray after it has become partially (and in some cases fully) occluded.

Figure 39 shows four sequential states of three objects as seen from one sensor. In the first frame, at time T0, none of the three objects is occluded. In the second frame at time T1, objects 1 and 2 have moved together. In the third frame, object 3 has moved behind objects 1 and 2. In the fourth frame, object 3 has begun to emerge from behind the left side of objects 1 and 2, but its right edge is still occluded.

The difference in the relation between objects 1 and 2, on the one hand, and between objects 2 and 3 will be considered. To represent the distinctions some new notation to the data-structure notation for raypaths shown in Figure 39 needs to be added. In frame 2, when objects 1 and 2 have moved together, their raypaths have been inserted for this sensor into

a cluster raypath node, represented by a circle with a C in it in the data structure diagram. The two partially-occluded objects are in a doubly-linked list attached to the cluster node. Below each occluded-raypath node in the data structure diagram are two letters. The letter on the left is called the edge-binding attribute of the raypath. This attribute tells us how to infer the angular position of the occluded edge(s) of a ray relative to the two edges of the cluster that are still visible. The cluster C always represents a ray (really a shadow) into which one or more others have merged. Therefore, it will always have two visible edges, even when the cluster overruns the edge of a sensor's field of vision.

The edge-binding attribute can have one of five states: (a) Unattached, (b) Left, (c) Right, (d) Interpolated, and (e) Velocity. The meaning of these states is as follows: Unbound is the initial state of an unoccluded ray. Left means that the raypath moves in concert with the left edge of the cluster. Whatever movement is made by the left edge of the cluster is also made by the left-edge-bound raypath. Right means that the raypath moves in concert with the right edge of the cluster. Whatever movement is made by the right edge of the cluster is also made by the right-edge-bound raypath. Interpolated means that the angular position of the raypath is interpolated between the left and right edges of the cluster. This mode will be elaborated further when we discuss object binding. Velocity means that the raypath is totally occluded and that it is not bound to an edge i.e. - the information is not available from the point of view of this sensor. In this case, we infer the raypath position by assuming that it will continue in uniform continuous motion (its past velocity is known) until we have data from other sensors to correct its position.

Returning to Figure 39, note what happens to the edge-binding attribute of each raypath as the raypath becomes occluded. At time T1 (frame 2), objects 1 and 2 have merged, so that the sensor sees a cluster consisting of the left side of raypath 1 and the right side of raypath 2. Therefore, the edge-binding attribute in the time T1 data structure diagram for raypaths 1 and 2 is L and R, respectively. The edge-binding attribute is the letter shown below the raypath node on the left side.

The letter on the right below each clustered raypath node is called the object-binding attribute. This attribute tells us, using knowledge from one or more other sensors, whether the two objects represented by two adjacent raypaths are in fact close enough together in the x-y plane to be attached, or travelling together. This knowledge is important, because if we know that two fingers, for example, are normally attached, we can, under certain conditions, make useful inferences about their location even when one or more fingers become totally occluded to all sensors.

This is what has happened at time T2 in Figure 39. The object represented by raypath 2 has become totally occluded. It cannot be seen from either the left or the right side of the cluster. Yet its position can be determined because it is known that it is attached to object 1, but not to object 3.

Object-binding attributes may have one of four states: (a) Unattached, (b) LeftObject, (c) RightObject, and (d) BothObjects. The meaning of these states is as follows: Unattached means that the object represented by the raypath is not attached to any other object. LeftObject means that the object represented by the raypath is attached to the object represented by the raypath immediately to the left. RightObject means that the object represented by the raypath is attached to the object represented by the raypath immediately to the right. BothObjects means that the object represented by the raypath is attached to the object represented by the raypath immediately to the left and to the object represented by the raypath immediately to the right. It is attached to both its neighbors.

Returning to the description of the edge and object-binding attributes of the raypath nodes, the attributes of the nodes progressing through the frames of Figure 39 have been "marked". At time T1, raypath or path 1 is bound to the left edge, and path 2 to the right edge of the cluster. Because it was found at the object level that paths 1 and 2 are very close to each other in the x-y plane, the object binding of path 1 is set at R, to indicate that it is "attached" to the object represented by the path on its right, namely path 2. Similarly, path 2's object binding is set to L, to indicate that it is attached to the object represented by the raypath on its left, namely path 1. We can say that path 1 and 2 are "attached".

At time T2, object 3 gets involved in the cluster by moving behind objects 1 and 2 from the right. From other sensors, it is known that it is not close to object 2. Also, note that path 2's right edge is now occluded. However, since it is attached to path 1, and not to path 3, its edge binding is changed to L, or left. This means that although one can no longer "see" path 2 from this sensor, we can infer its position relative to path 1, to which it is bound, and from this, its position relative to the left edge of the cluster.

At time T2, the edge-binding of path 3 is set to R (right), but its object-binding attribute is set to U, or Unattached.

At time T3, path 3 passes behind paths 1 and 2, and reappears on the left. Its reappearance at the left side of the cluster can be predicted and detected if its velocity remains relatively constant after it becomes completely occluded. These examples have been provided in order to clarify the role of edge and object-binding attributes in inferring the position of occluded objects. With an understanding of the edge and object binding attributes, one can gain enhanced understanding of the MatchRay function 178 (Fig. 22) in greater detail.

Figure 40 illustrates the MatchRay function 220. When the function is called, it returns 224 if the raw raylist is empty 222. If the current raypath is not a new one 226, it infers the present raypath position from knowledge of its previous position and velocity 228.

Operations 230 through 264 in Figure 40 comprise a loop which attempts to find overlap between the current inferred position of the raypath and each raw

ray in the current raylist for the current sensor. (The raw rays represent updated position information for each raypath.)

For each raw ray in turn 230, it: (1) rejects it if it is too small or too large 234; (2) compares it with the current raypath to see if there is overlap 238; (3) if there is overlap, and if the current raypath also overlapped the previous raw ray 250, then the ray has split, and if the raypath is a cluster it must be divided into two discrete clusters by the SplitRays function 252; (4) if it is found 256, that the current raw ray (in the raylist) also matched the previous raypath for this sensor, then the current and previous raypaths have merged; if the current and previous raypaths are not bound to the same object, we can assume that two distinct objects have merged from the point of view of the current sensor 258 and 260, and the MergeRayPaths function is called 262; (5) if there is no overlap between the current raypath and the current raw ray 238, the next raw ray is fetched. If the last raw ray has been examined 264, MatchRay returns 266.

Referring to Figures 41-43, the MergeRayPaths function setting of edge and object binding attributes during the merging of rays is what permits the ray-tracking function to infer the positions of the occluded rays and therefore the positions of occluded and partially-occluded objects. The MergeRayPaths function is included in its entirety as it is important in determining the effectiveness of the Sensor Frame in tracking occluded objects.

MergeRayPaths can be divided into two logically independent parts. The first part, from portion 290 in Figure 41, up to but not including portion 310, is concerned only with the mechanisms of merging the data structures of each of the two merging raypaths or raypath clusters into one cluster, and is readily understood by one skilled in the art.

The second part of MergeRayPaths from portion 310 through portions 354, 388 and 408 in Figures 42a, 42b and 43, respectively, contain the ray and edge-binding algorithms employed in the operation of a peripheral-scan multi-object scanning device such as the Sensor Frame device.

If the raypaths to be merged are raypaths of objects which are close together in the x-y plane 310, one then determines whether the left raypath's object was previously unattached 320. If it was attached, one then determines whether the right path's object was previously unattached 376. If it was attached, then both objects are to be attached in both directions 378. If the right path's object was unattached, it is bound to the left path's object, and the left path's object is bound in both directions 380. If the right path is occluded 372, both the right and left paths have their edge-binding attribute set to "BindToLeftEdge" 384. Otherwise, as the right path is unoccluded, its edge-binding is set to "BindToRightEdge", and the left path's edge-binding is set to "Interpolate" 386.

If at 320, the left raypath's object was found to be previously unattached, one determines whether the right raypath's object was unattached 324. If it was attached, then the right path's object is bound in both directions, and the left path's object is bound to the right path's object 350. If the left path is unoccluded 351, its edge binding is set to "BindToLeftEdge", and the right path's edge binding is set to "Interpolate" 353. If the left path is occluded, both paths' edge binding is set to "BindToRightEdge" 352.

If at 324, the right raypath's object was unattached, then the objects of both raypaths were previously unattached, and the left object is attached (bound) to the right and vice-versa 330. One then determines whether the left path was previously unoccluded 332. If so, the left path edge binding is set to "BindToLeftEdge" 334. One then determines whether the right path was occluded 360. If it was, its edge binding is set to "BindToLeftEdge", otherwise it is set to "BindToRightEdge".

If at 332, it was determined that the left path was occluded, one determines whether the right path is also occluded 366. If it is, the edge binding of both the left and right raypaths is set to "Velocity" since they are both completely occluded and unattached to anything but each other.370. If the right path is not occluded, the edge-binding of both paths is set to "BindToRightEdge".

If at 310, it was found that the objects represented by the left and right raypaths were not close together in the x-y plane, one then determines whether the left path is already attached 400. If it is, its edge-binding is set to "BindToLeftEdge" 402. If not, one determines whether the left path is in a cluster 416. If so, the left path edge-binding is set to "Velocity" because it is completely occluded and unattached 420. If not, its edge-binding is set to "BindToLeftEdge" 422.

One then determines whether the right raypath is already attached 404. If it is, the right path edge-binding is set to "BindToRightEdge" 406. If not, one determines whether the right path is in a cluster 410. If it is, its edge-binding is set to "Velocity" because it is completely occluded and unattached 412. If not, its edge-binding is set to "BindToRightEdge" 414. The function returns at 408.

It will be appreciated that the sensors will emit an electrical signal corresponding to the individual cells of the sensor array which were recipients of light and that a suitable controller (Figure 21) will relate the signal to the specific position and, if desired, velocity of the light occluding objects and may isolate certain graphic objects based upon such occlusion. Velocity would be determined by comparing position changes of the objects as related to time.

It will be appreciated that the method and apparatus of the present invention may involve providing a video monitor, an adjacent frame, a plurality of optical sensors generally facing the frame interior and illuminating at least a portion of the space defined by the frame. A plurality of light occluding objects are employed to obstruct a portion of the light from reaching certain cells of the sensors. Signals are emitted from the sensors corresponding to the three-dimensional positions of the light occluding objects with respect to the frame.

It will be appreciated that, while a suitable protective dust cover could be placed over the frame when it is not in use to resist undesired entry of

foreign matter, such frame has not been illustrated in the drawings.

While reference has been made herein to use of multiple fingers within the frame, it will be appreciated that the invention is not so limited and that feet, wands or other physical objects or combinations thereof may be employed effectively.

It will be appreciated that while the disclosure has focused on examples of use of a single frame, it will be appreciated that a number of frames may be employed in adjacent areas if desired.

It will be appreciated that while primary reference has been made herein to modification of graphic objects and that a wide variety of uses such as in connection with computers generally, musical keyboards, computer aided design systems and safety devices for industrial equipment will become apparent to those skilled in the art, numerous other uses may exist.

In a preferred approach to the invention, the scanning of the visual field by means of the sensors can be effected rapidly on the order of about 30 to 60 times per second, for example.

It will be appreciated that while for convenience of illustration a frame which is generally rectangular has been shown, the invention is not so limited. It is preferred that the frame be a closed figure or substantially closed, that it be provided with a suitable number of sensors with four being the preferred minimum number (although two sensors may be employed in certain specialized situations) and there being no maximum and that the light illumination be substantially uniform. It will be appreciated that the frame may be square, circular or of any other desired shape so long as the light field and associated sensors will permit effective determination of position and/or shape of a light occluding object and that determination of velocity, if desired, is permitted.

It will be appreciated that while for clarity of disclosure reference has been made herein to positioning of the frame adjacent to a video monitor, such positioning is not required. The frame may be used without the video monitor, if desired.

While certain examples of use of the invention have been provided a large number of additional uses will be apparent to those skilled in the art. For example, the system may be employed to play a digital musical organ by permitting the output of the sensors to operate the organ. Also, the system is compatible with speech activated computer operations. This facilitates flexibility of use of the system.

It will be apparent to those skilled in the art that there are numerous end use applications of the method and apparatus of this invention. A basic use would be for gesture sensing devices in computer based workstations. The invention could be used in industrial processes and in safety devices in an industrial environment. The system could also be used in robotic systems and systems for the handicapped, for example.

It will be appreciated, therefore, that the present invention provides a method and apparatus for permitting efficient selection and manipulation of graphic objects through determination of the position and velocity of a plurality of light occluding objects by peripheral scan means. This is accomplished through the use of a video monitor, a frame having sources of illumination, optical sensors at predetermined positions and processing of the signals emerging from said sensors.

Whereas particular embodiments of the invention have been disclosed for purposes of illustration, it will be evident to those skilled in the art that numerous variations of the details may be made without departing from the invention as defined in the appended claims.

The peripheral scan apparatus disclosed with reference to the drawings will quickly and efficiently determine the position, angle or orientation and velocity of multiple objects introduced into the frame area, and thereby permits efficient manipulation of graphic objects, such as translation, rotation, and scaling.

The method and apparatus described provide for manipulation of position and orientation of graphic objects or components thereof, so as to permit composition and decomposition (assembly and disassembly) of such graphic objects.

The system described furthermore provides a real-time determination of sensed-object position, orientation and velocity and which is economical and easy to employ.

The features disclosed in the foregoing description, in the following claims and/or in the accompanying drawings may, both separately and in any combination thereof, be material for realising the invention in diverse forms thereof.

Claims

1. A method of manipulating graphic objects comprising providing a frame, and a plurality of optical sensors generally facing the frame interior, illuminating at least a portion of the space defined by said frame, employing a plurality of light occluding objects to obstruct a portion of the light from reaching at least certain elements of said sensor, and emitting signals from said sensors corresponding to the positions of said light occluding objects with respect to said frame.

2. The method of claim 1 including periodically combining said sensor signals to provide information regarding changes in position of said light occluding objects, storing a first set of said sensor signals corresponding to a first position of said light occluding objects and obtained at a first time, and subsequently comparing said first set of sensor signals with a second set of said sensor signals obtained at a second time to determine changes in position of said light occluding objects.

3. The method of claim 2 including effecting a comparison of the position and relative motion of the edges and visual centers of said graphic objects, employing said comparison in maintaining a history of the positions, orientations

and motion of said objects, and employing said history in resolving ambiguous combinations of said objects or in tracking objects at least partially occluded by other said objects.

4. The method of claim 2 including comparing a plurality of sets of said sensor signals with each being obtained at a different time, and employing said sets to determine changes in position of said light occluding objects.

5. The method of claim 2 including employing portions of the human body as said light occluding objects.

6. The method of claim 2 including providing a video monitor, providing said frame as a closed frame, positioning said frame adjacent to said video monitor, illuminating the interior of said frame substantially uniformly, and employing dynamic RAM sensors as said sensors.

7. The method of claim 5 including employing said light occluding objects to select and manipulate at least one graphic object.

8. Apparatus for manipulating graphic objects comprising frame means for at least partially receiving light occluding objects, light source means at least partially illuminating the region adjacent said frame means, sensor means operatively associated with said frame means for receiving at least a portion of said illumination and emitting responsive signals, and signal processing means for employing said sensor signals to determine the position of said light occluding objects.

9. The apparatus of claim 8 including a video monitor disposed adjacent to said frame means, said frame means being disposed in front of said video monitor with at least a portion of said monitor being visible through said frame.

10. The apparatus of claim 8 wherein said signal processing means has means for comparing a first set of said sensor signals taken at a first time with a second set of signals taken subsequently in order to determine changes in position of said light occluding objects.

11. The apparatus of claim 8 wherein said sensor means has a plurality of dynamic RAM sensors, said signal processing means having means for storing and comparing a plurality of sensor signal sets to determine changes in position in said light occluding objects, and said signal processing means having means for manipulating said graphic objects responsive to the information received from said sensors.

12. The apparatus of claim 11 wherein said frame means is generally rectangular, and said dynamic RAM sensors are disposed generally in the corners thereof.

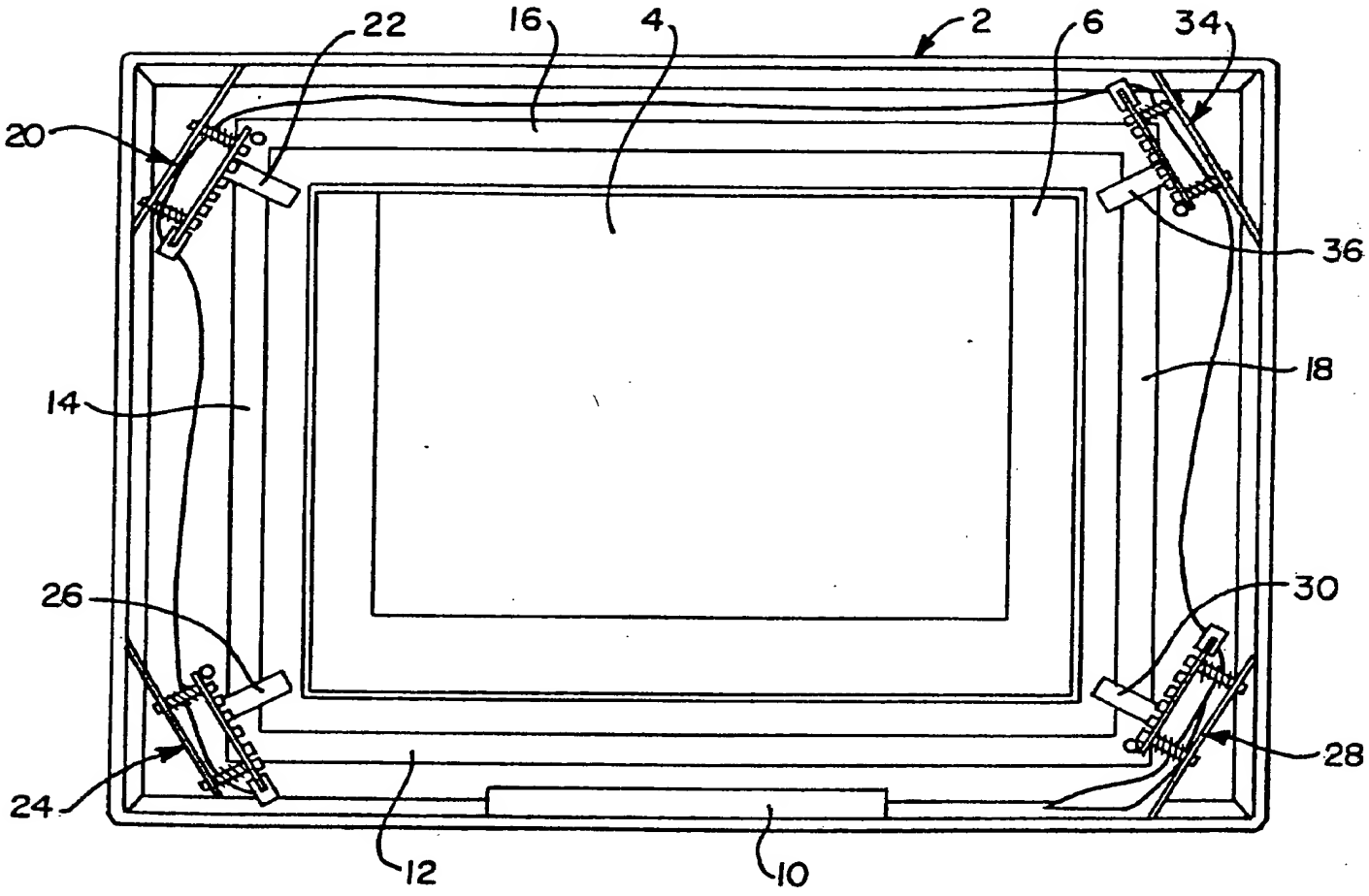

FIG. 1

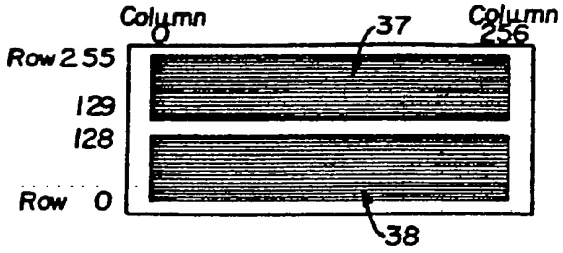

FIG. 2

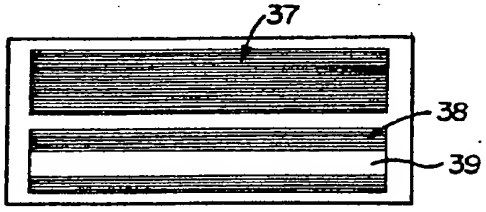

FIG. 3

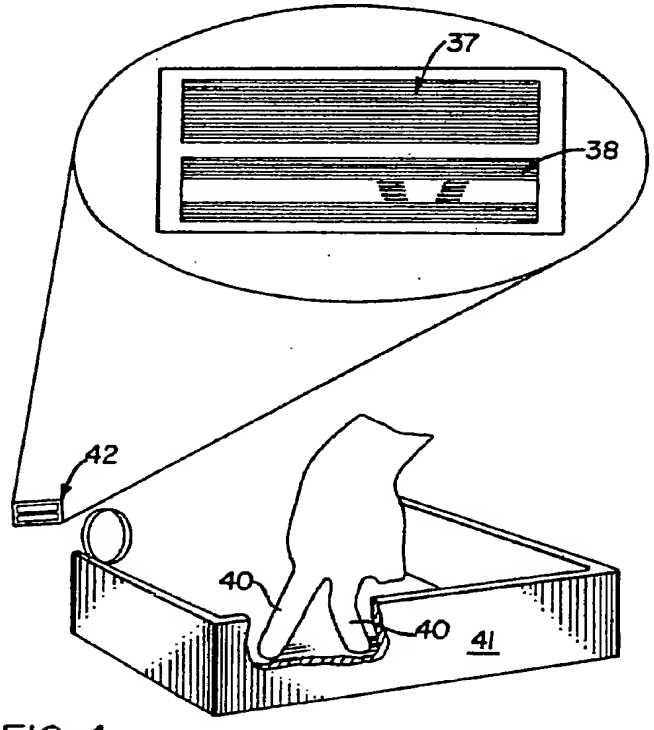

FIG. 4

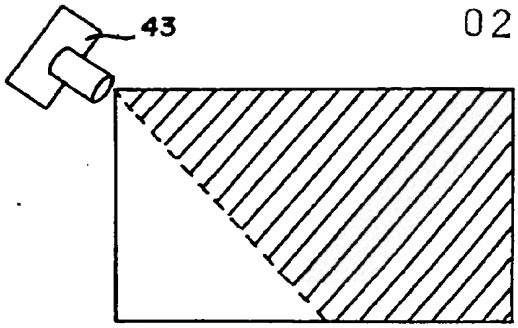

FIG.5

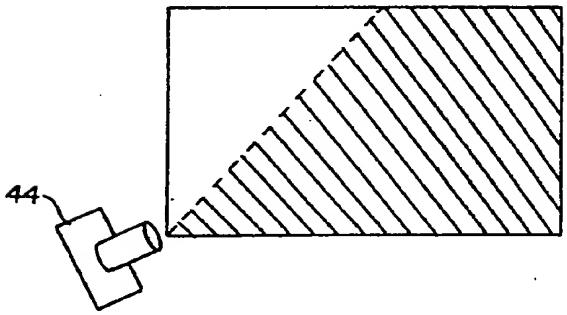

FIG. 6

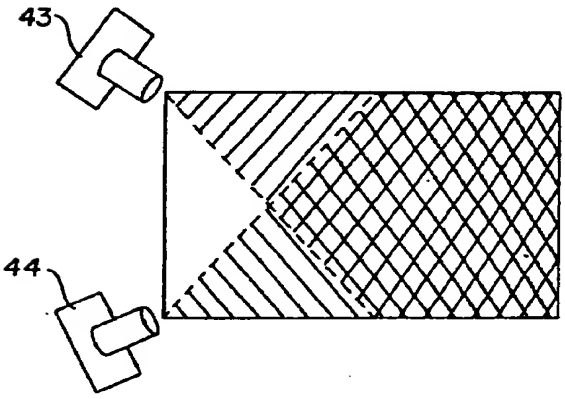

FIG. 7

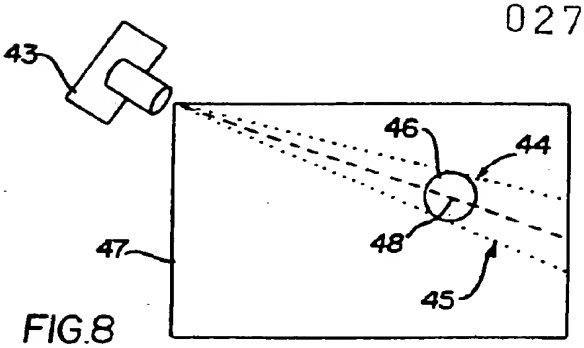

FIG.8

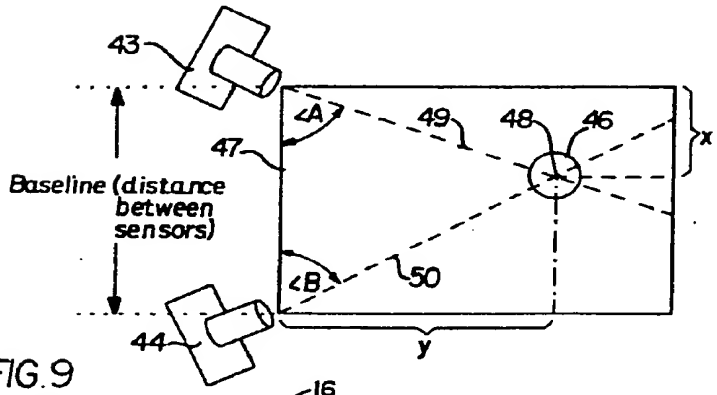

FIG.9

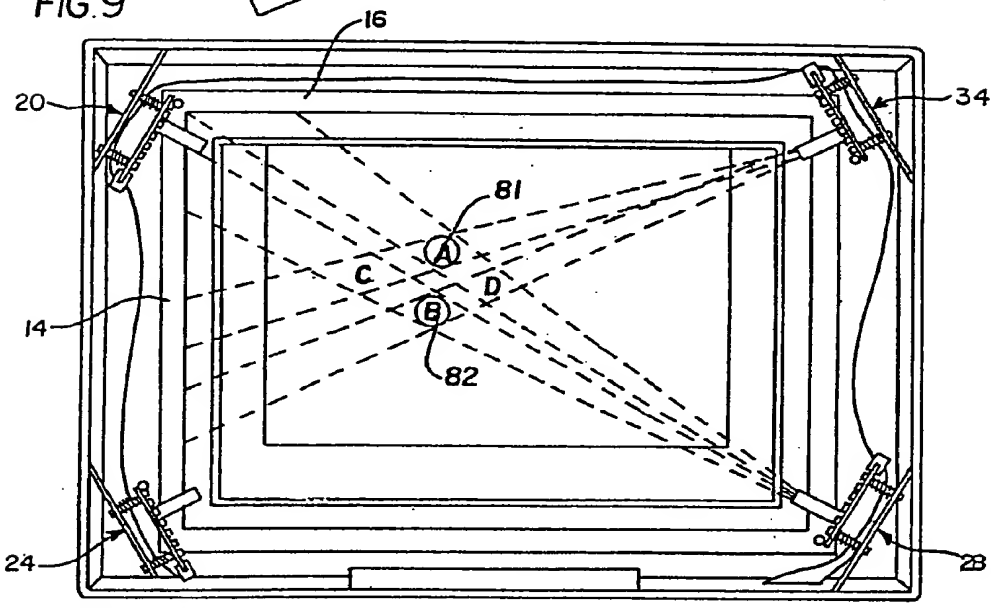

FIG 10

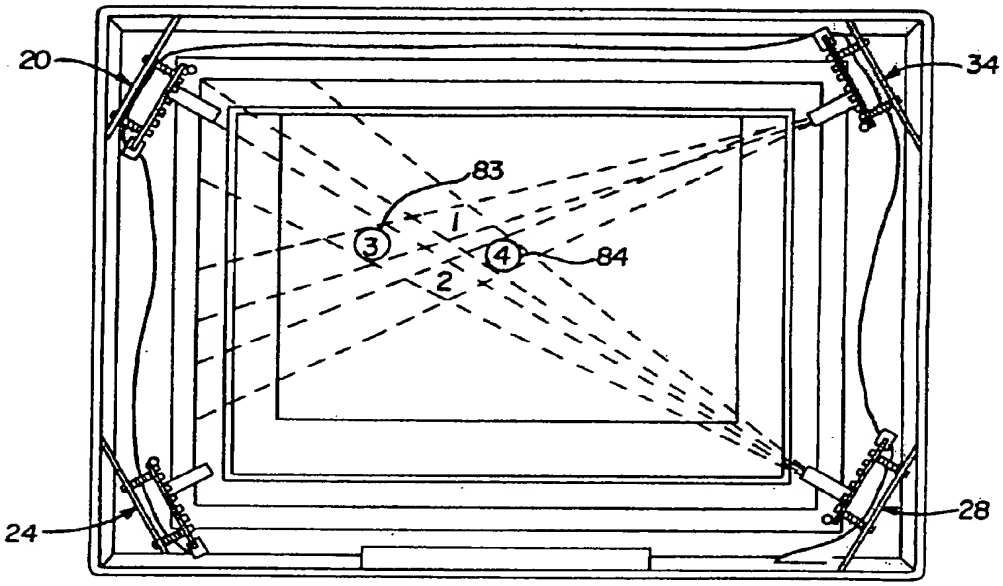

FIG. 11

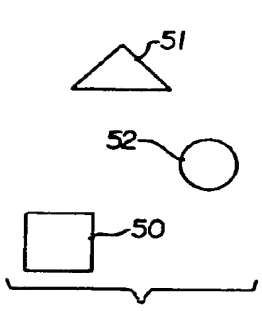

FIG. 12a

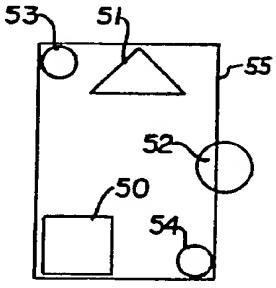

FIG. 12b

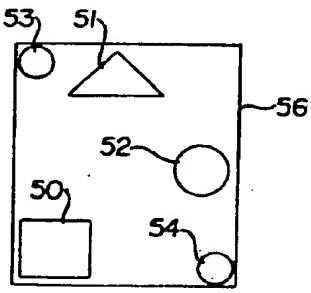

FIG. 12c

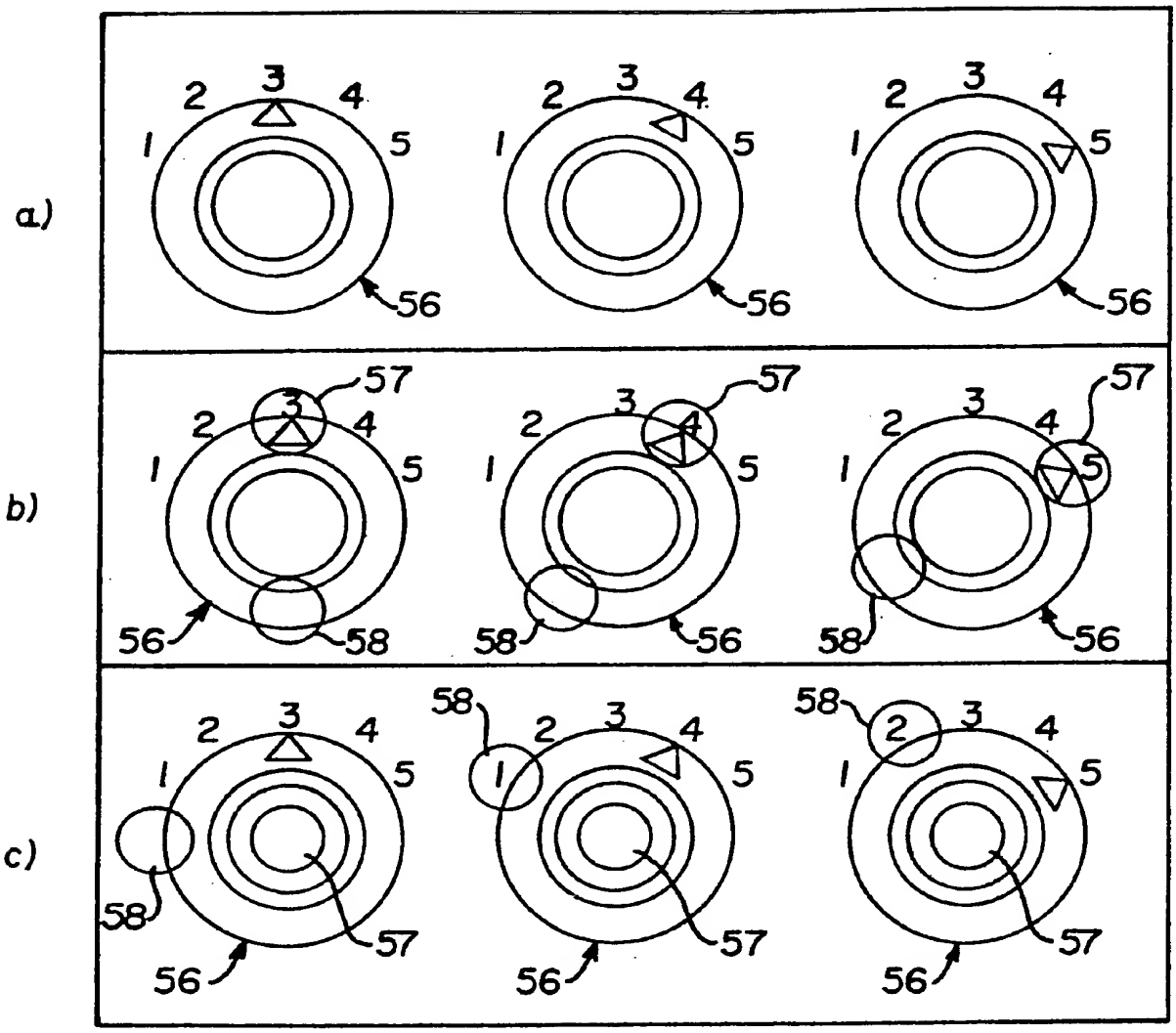

FIG. 13

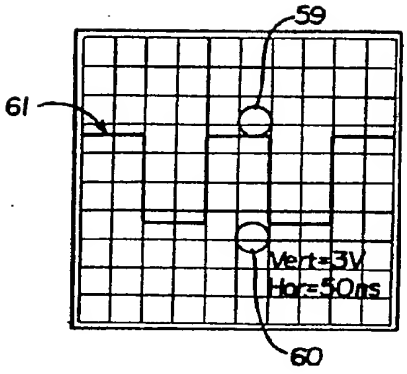

FIG. 14a

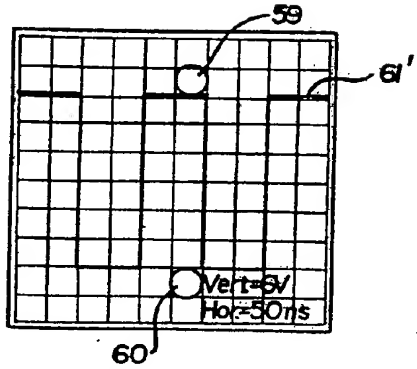

FIG. 14b

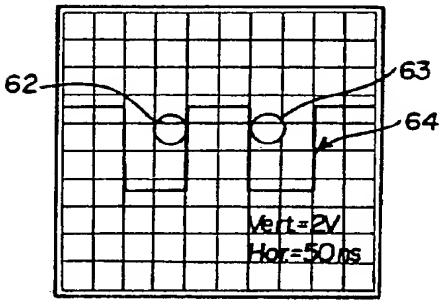

FIG. 15a

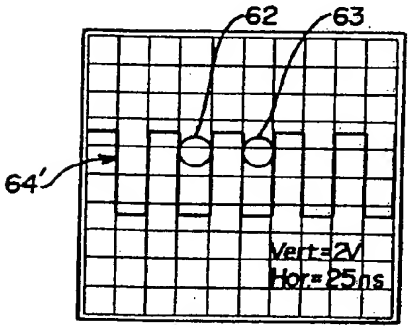

FIG. 15b

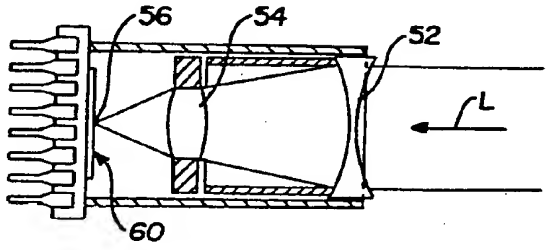

FIG. 16

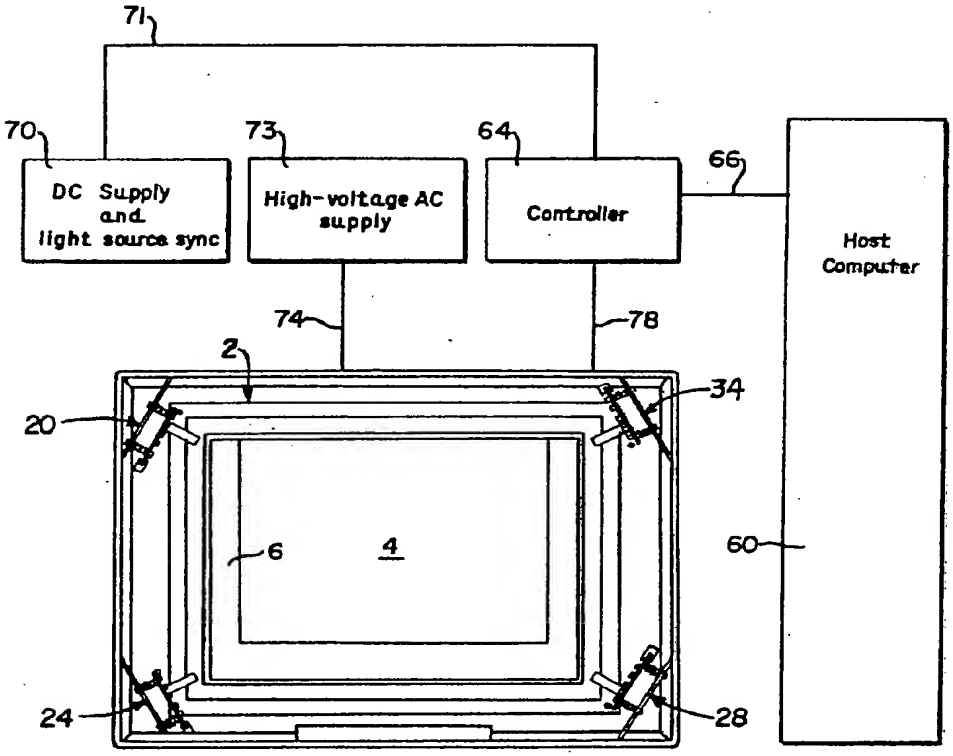
71
70
73
64
66
DC Supply and light source sync
High-voltage AC supply
Controller
Host Computer
74
78
2
20
34
6
4
60
24
28

FIG. 17

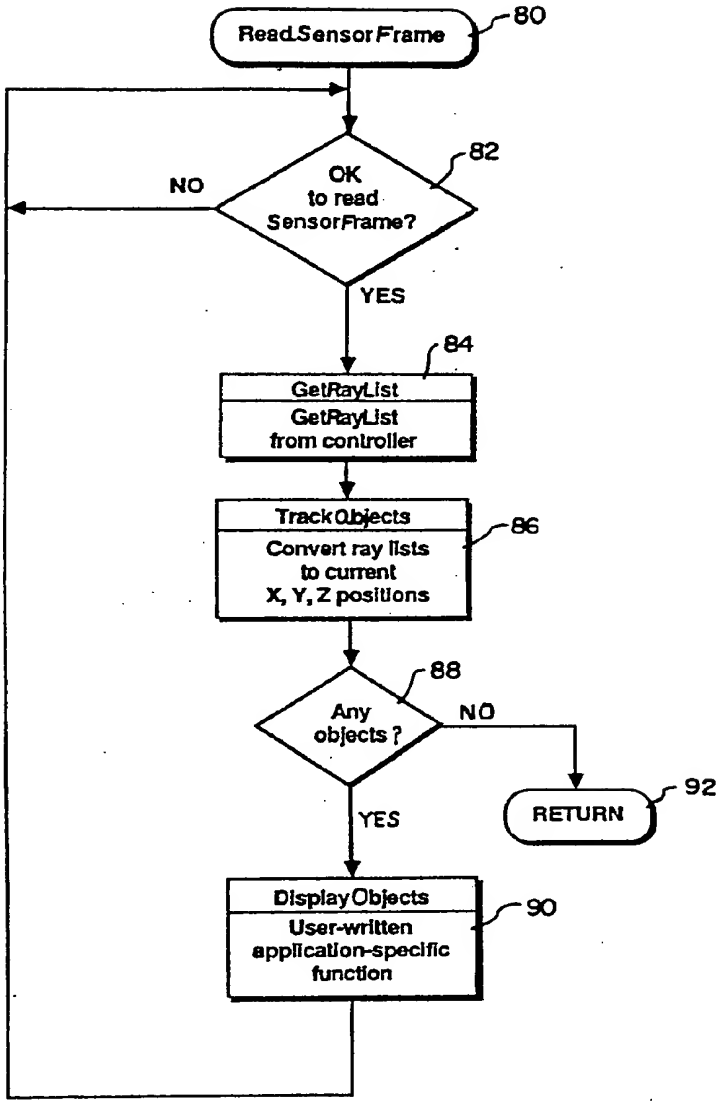
ReadSensorFrame
80
OK to read SensorFrame?
82
NO
YES
84
GetRayList
GetRayList from controller
TrackObjects
Convert ray lists to current X, Y, Z positions
86
Any objects ?
88
NO
YES
RETURN
92
DisplayObjects
User-written application-specific function
90

FIG. 18

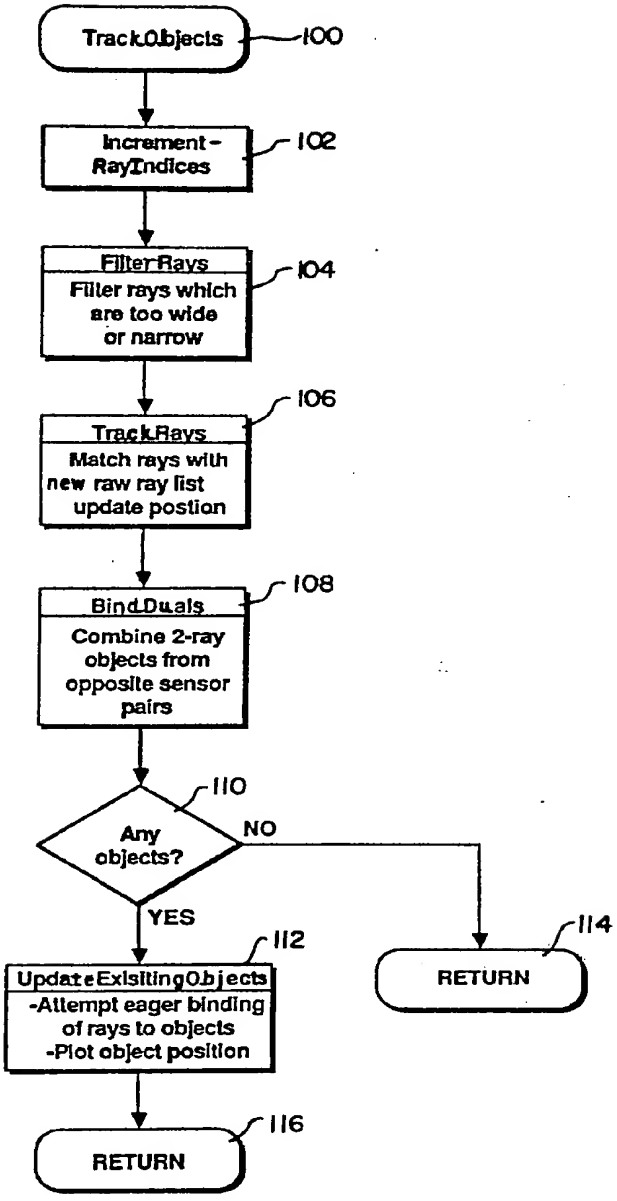
TrackObjects
100
Increment-RayIndices
102
FilterRays
Filter rays which are too wide or narrow
104
TrackRays
Match rays with new raw ray list update postion
106
BindDuals
Combine 2-ray objects from opposite sensor pairs
108
Any objects?
110
NO
YES
UpdateExisitingObjects
-Attempt eager binding of rays to objects
-Plot object position
112
RETURN
114
RETURN
116

FIG. 19

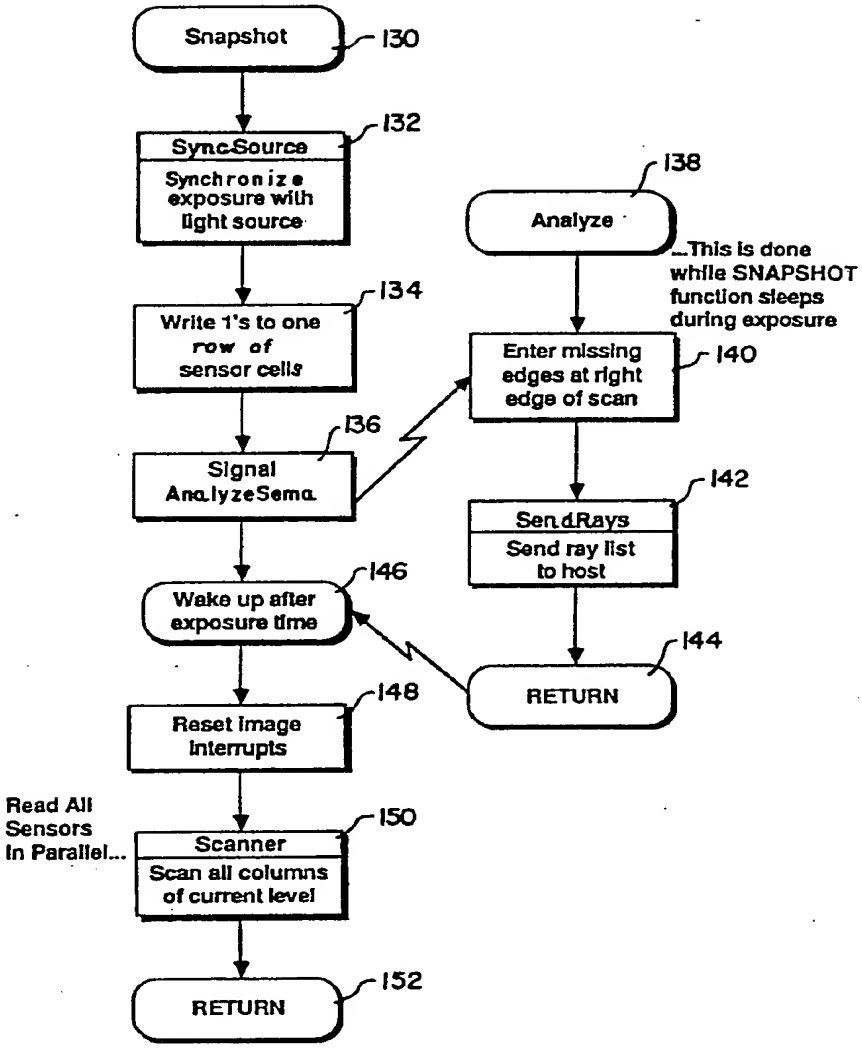
Snapshot
130
SyncSource
Synchronize exposure with light source
132
Write 1's to one row of sensor cells
134
Signal AnalyzeSema
136
Wake up after exposure time
146
Reset Image Interrupts
148
Read All Sensors In Parallel...
Scanner
Scan all columns of current level
150
RETURN
152
Analyze
138
...This is done while SNAPSHOT function sleeps during exposure
Enter missing edges at right edge of scan
140
SendRays
Send ray list to host
142
RETURN
144

FIG. 20

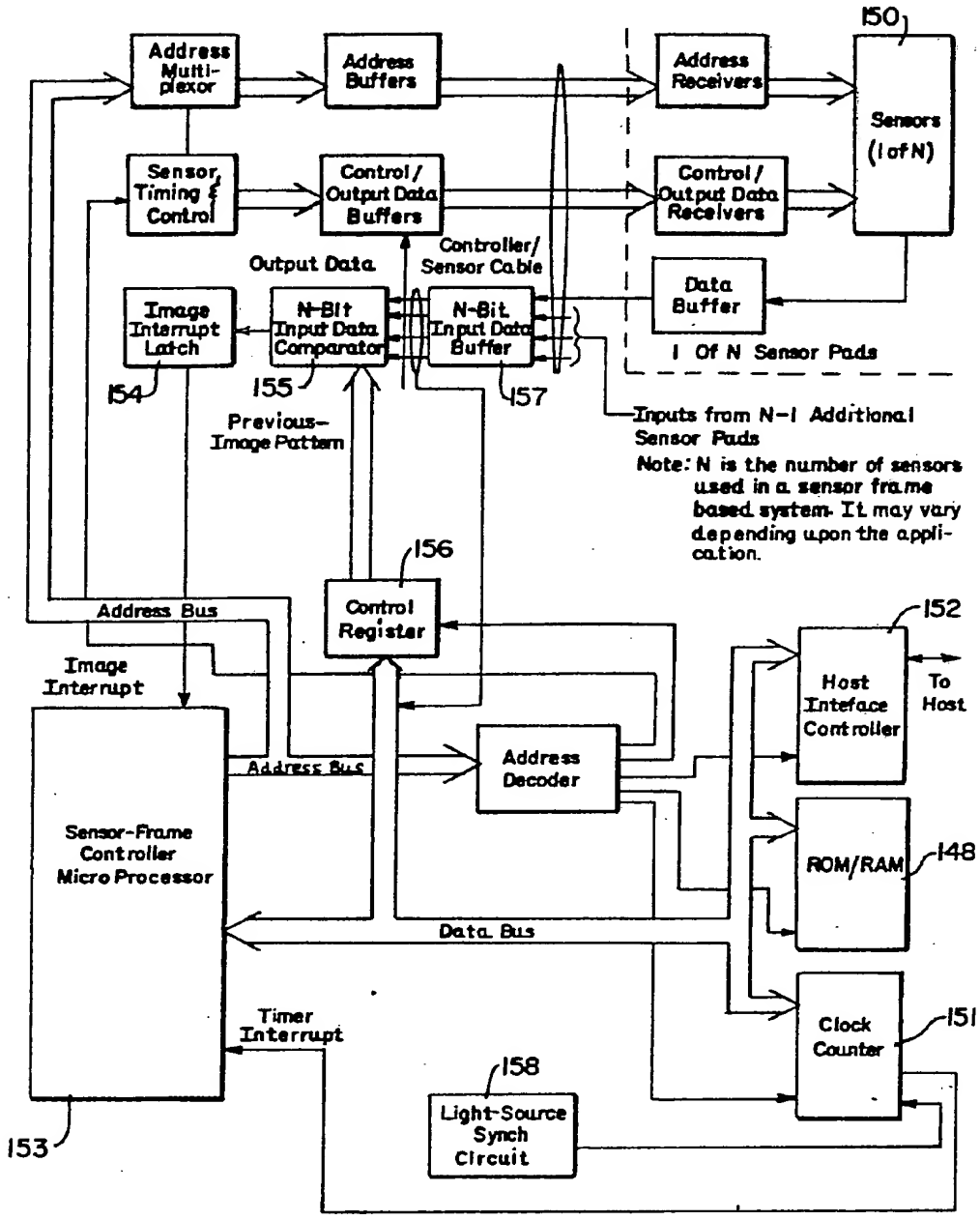
150
Address Multi-plexor
Address Buffers
Address Receivers
Sensors (1 of N)
Sensor Timing & Control
Control / Output Data Buffers
Control / Output Data Receivers
Output Data
Controller/ Sensor Cable
Data Buffer
Image Interrupt Latch
N-Bit Input Data Comparator
N-Bit Input Data Buffer
154
155
157
1 Of N Sensor Pads
Previous-Image Pattern
Inputs from N-1 Additional Sensor Pads
Note: N is the number of sensors used in a sensor frame based system. It may vary depending upon the application.
156
Control Register
152
Address Bus
Image Interrupt
Host Inteface Controller
To Host
Address Bus
Address Decoder
Sensor-Frame Controller Micro Processor
ROM/RAM
148
Data Bus
Timer Interrupt
Clock Counter
151
158
Light-Source Synch Circuit
153

FIG. 21

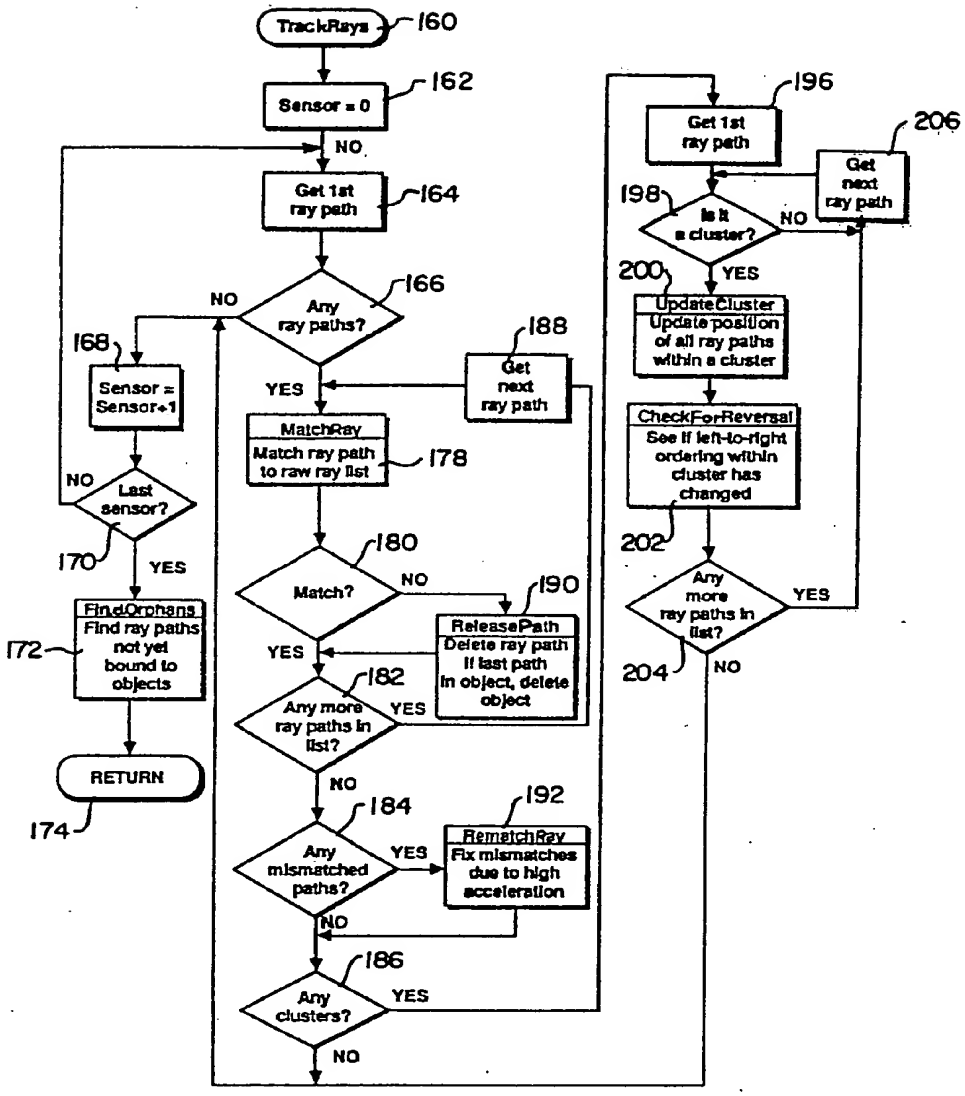
TrackRays
160
Sensor = 0
162
NO
Get 1st ray path
164
Any ray paths?
166
NO
YES
168
Sensor = Sensor+1
NO
Last sensor?
170
YES
FindOrphans
Find ray paths not yet bound to objects
172
RETURN
174
Get next ray path
188
MatchRay
Match ray path to raw ray list
178
Match?
180
NO
YES
ReleasePath
Delete ray path If last path in object, delete object
190
Any more ray paths in list?
182
YES
NO
Any mismatched paths?
184
YES
RematchRay
Fix mismatches due to high acceleration
192
NO
Any clusters?
186
YES
NO
Get 1st ray path
196
Get next ray path
206
Is it a cluster?
198
NO
YES
UpdateCluster
Update position of all ray paths within a cluster
200
CheckForReversal
See if left-to-right ordering within cluster has changed
202
Any more ray paths in list?
204
YES
NO

FIG. 22

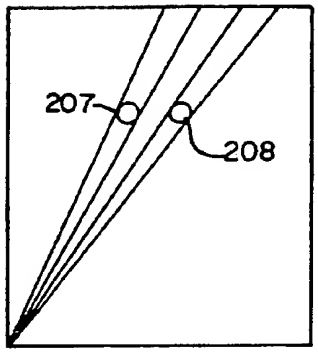

FIG. 23

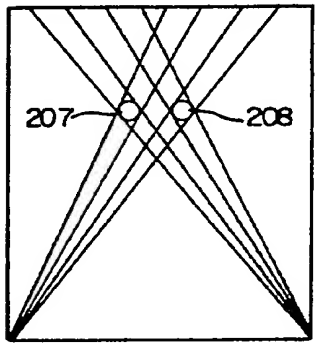

FIG. 24

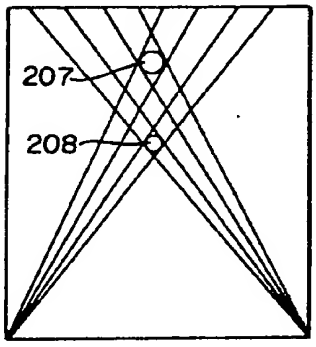

FIG. 25

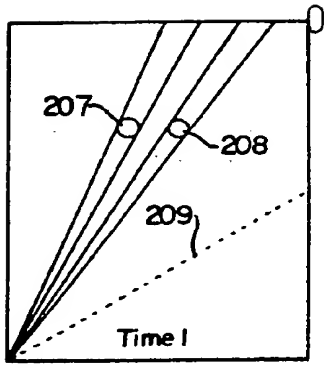

FIG. 26

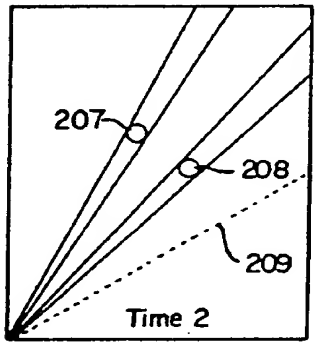

FIG. 27

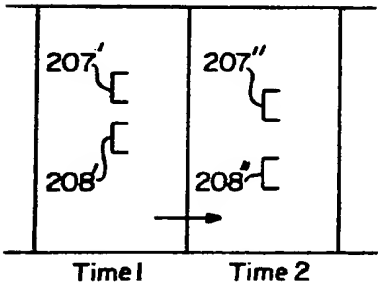

FIG. 28

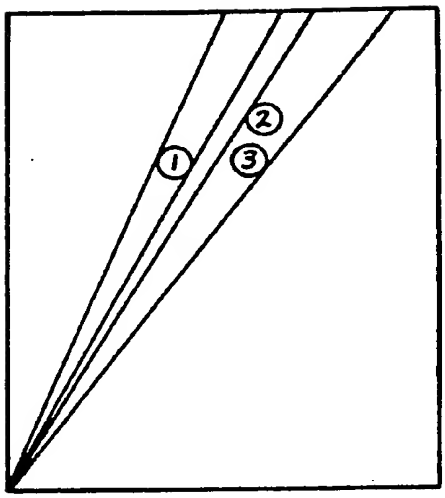

FIG. 29

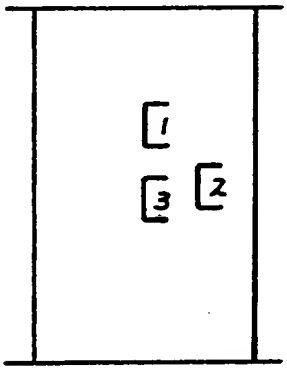

FIG. 30

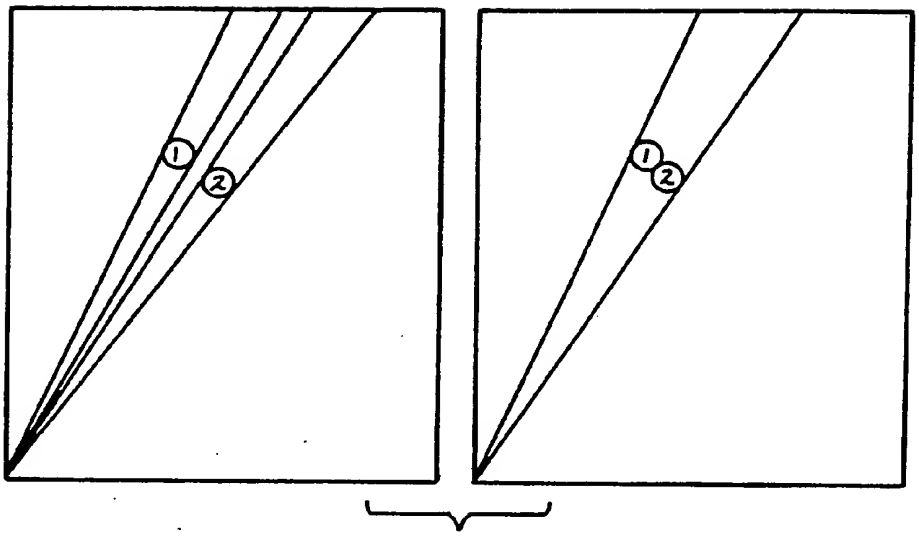

FIG. 31

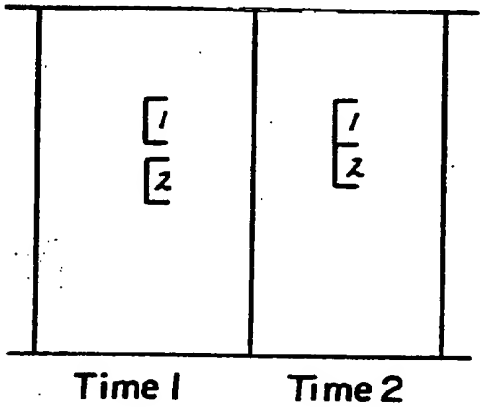

FIG. 32

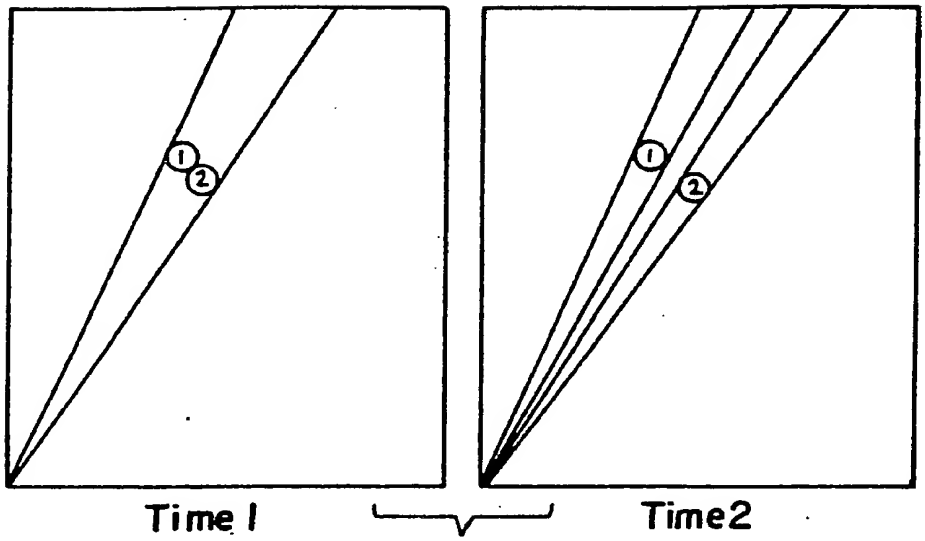

FIG.33

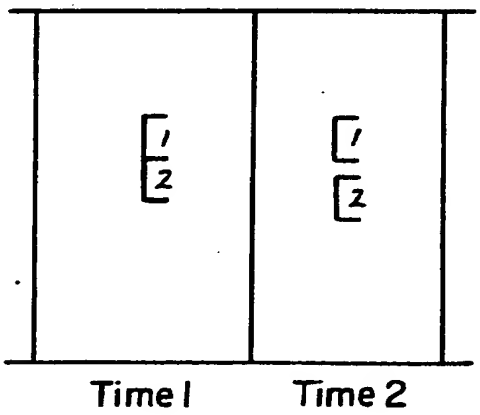

FIG. 34

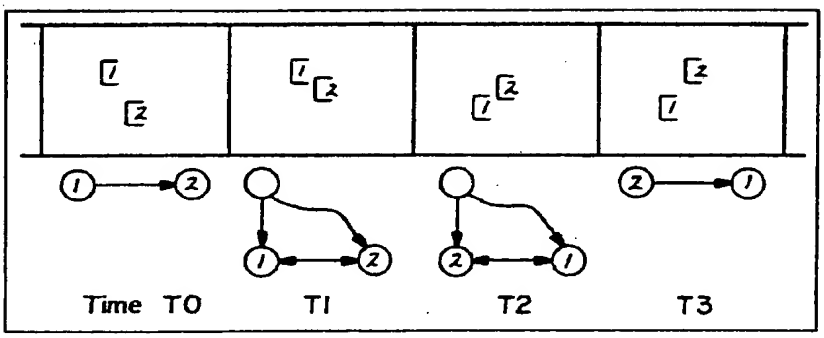

FIG. 35

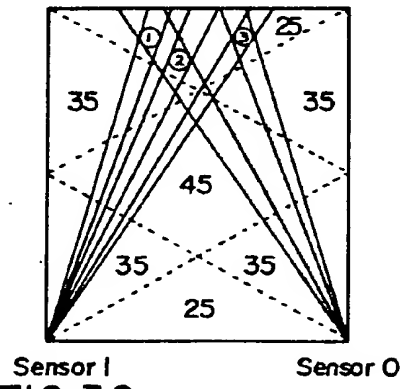

FIG. 36

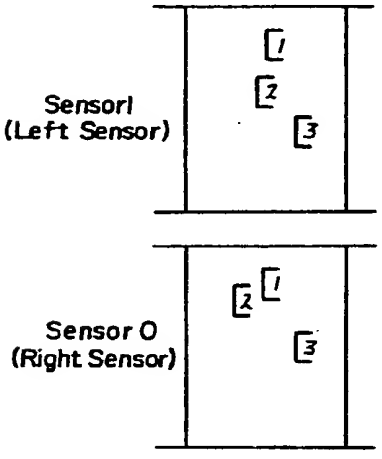

FIG. 37

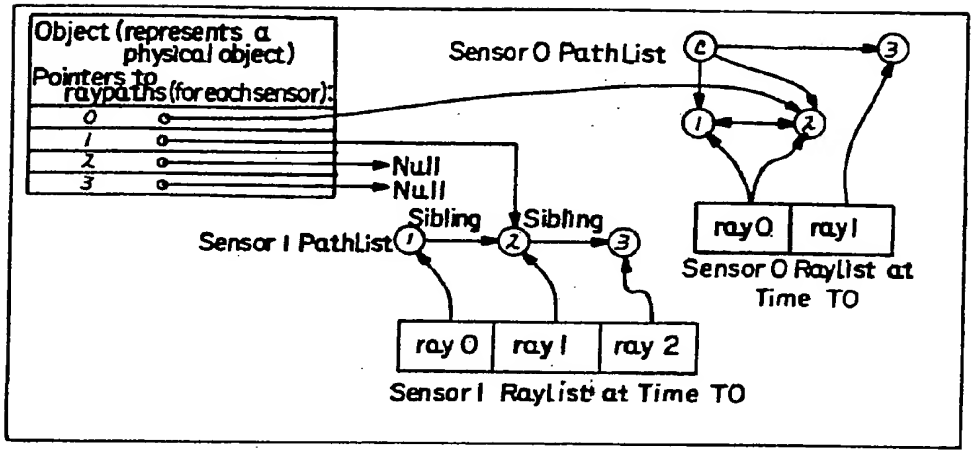
Object (represents a physical object)
Pointers to raypaths (for each sensor):
0
1
2
3
Null
Null
Sensor 0 PathList
Sibling
Sibling
Sensor 1 PathList
ray 0
ray 1
Sensor 0 Raylist at Time T0
ray 0
ray 1
ray 2
Sensor 1 Raylist at Time T0

FIG. 38

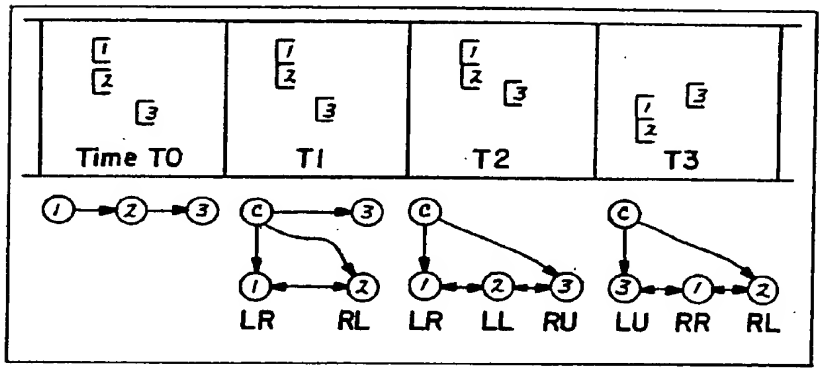
Time T0
T1
T2
T3
LR
RL
LR
LL
RU
LU
RR
RL

FIG. 39

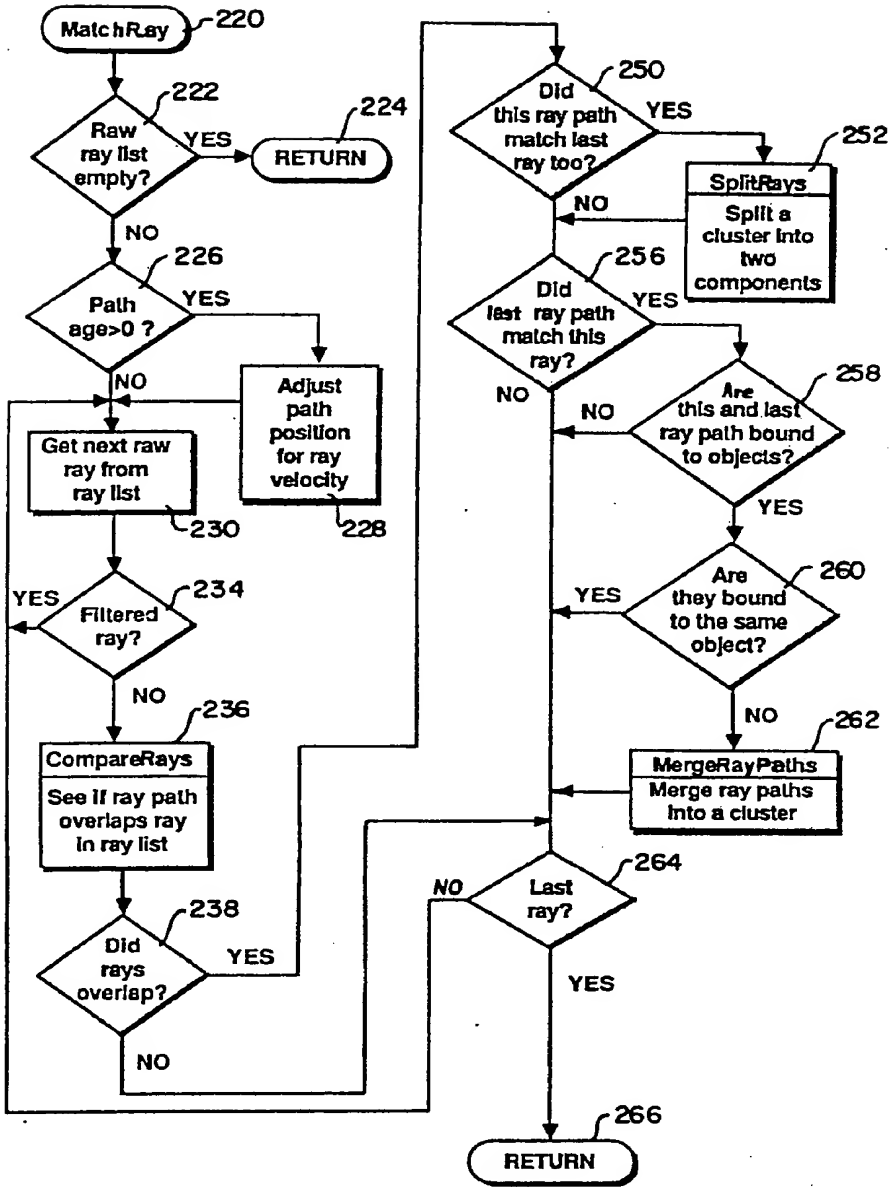
MatchRay
220
Raw ray list empty?
222
YES
RETURN
224
NO
Path age>0 ?
226
YES
NO
Adjust path position for ray velocity
228
Get next raw ray from ray list
230
YES
Filtered ray?
234
NO
CompareRays
See if ray path overlaps ray in ray list
236
Did rays overlap?
238
YES
NO
Did this ray path match last ray too?
250
YES
SplitRays
Split a cluster into two components
252
NO
Did last ray path match this ray?
256
YES
NO
Are this and last ray path bound to objects?
258
NO
YES
Are they bound to the same object?
260
YES
NO
MergeRayPaths
Merge ray paths into a cluster
262
Last ray?
264
NO
YES
RETURN
266

FIG. 40

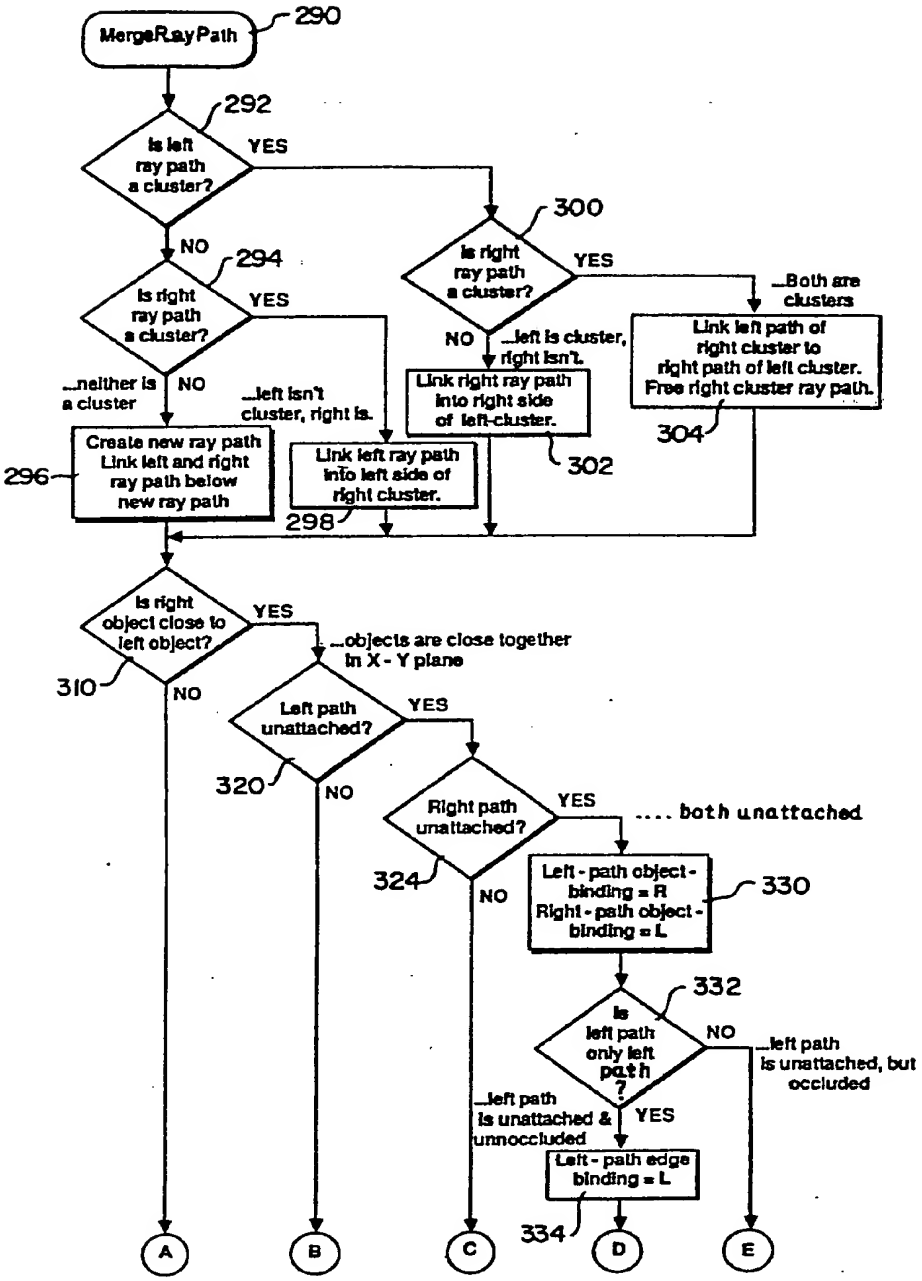
MergeRayPath
290
292
Is left ray path a cluster?
YES
NO
294
Is right ray path a cluster?
YES
NO
...neither is a cluster
...left isn't cluster, right is.
Create new ray path Link left and right ray path below new ray path
296
Link left ray path into left side of right cluster.
298
300
Is right ray path a cluster?
YES
NO
...left is cluster, right isn't.
Link right ray path into right side of left-cluster.
302
...Both are clusters
Link left path of right cluster to right path of left cluster. Free right cluster ray path.
304
Is right object close to left object?
YES
NO
310
...objects are close together in X - Y plane
Left path unattached?
YES
NO
320
Right path unattached?
YES
NO
324
.... both unattached
Left - path object - binding = R
Right - path object - binding = L
330
Is left path only left path ?
332
NO
YES
...left path is unattached, but occluded
...left path is unattached & unnoccluded
Left - path edge binding = L
334
A
B
C
D
E

FIG. 41

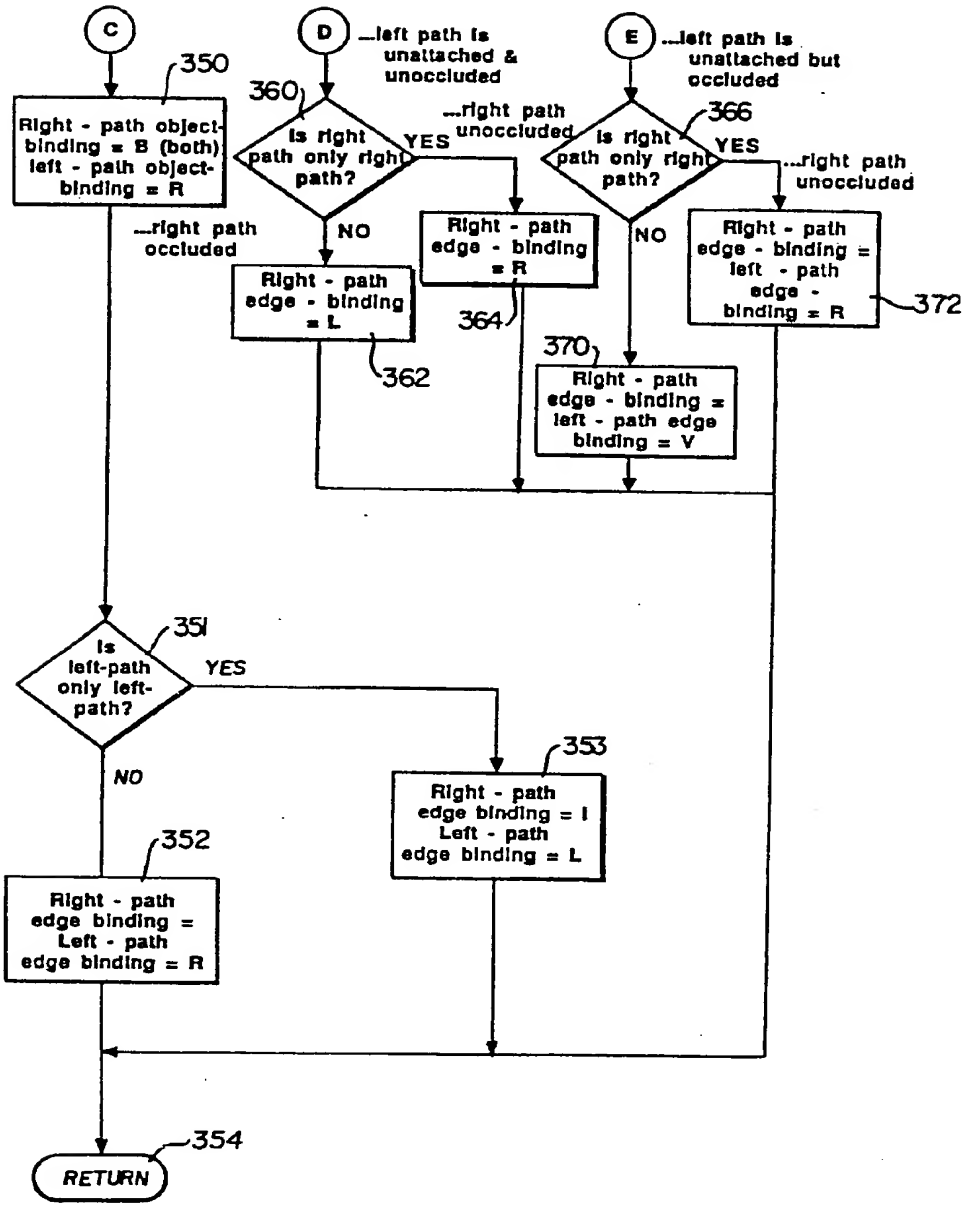
C
D
...left path is unattached & unoccluded
E
...left path is unattached but occluded
350
Right - path object-binding = B (both) left - path object-binding = R
360
Is right path only right path?
YES
...right path unoccluded
NO
...right path occluded
366
Is right path only right path?
YES
...right path unoccluded
NO
Right - path edge - binding = L
362
Right - path edge - binding = R
364
Right - path edge - binding = left - path edge - binding = R
372
370
Right - path edge - binding = left - path edge binding = V
351
Is left-path only left-path?
YES
NO
353
Right - path edge binding = I Left - path edge binding = L
352
Right - path edge binding = Left - path edge binding = R
354
RETURN

FIG. 42 a

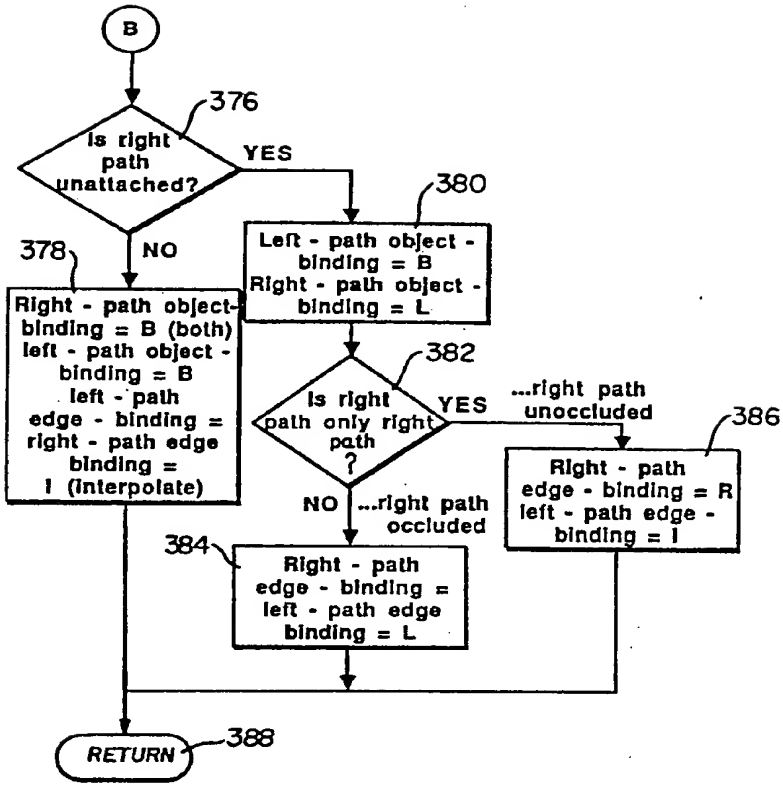
B
376
Is right path unattached?
YES
NO
380
Left - path object - binding = B
Right - path object - binding = L
378
Right - path object - binding = B (both)
left - path object - binding = B
left - path edge - binding = right - path edge binding = I (interpolate)
382
Is right path only right path ?
YES
...right path unoccluded
386
Right - path edge - binding = R
left - path edge - binding = I
NO
...right path occluded
384
Right - path edge - binding = left - path edge binding = L
RETURN
388

FIG. 42b

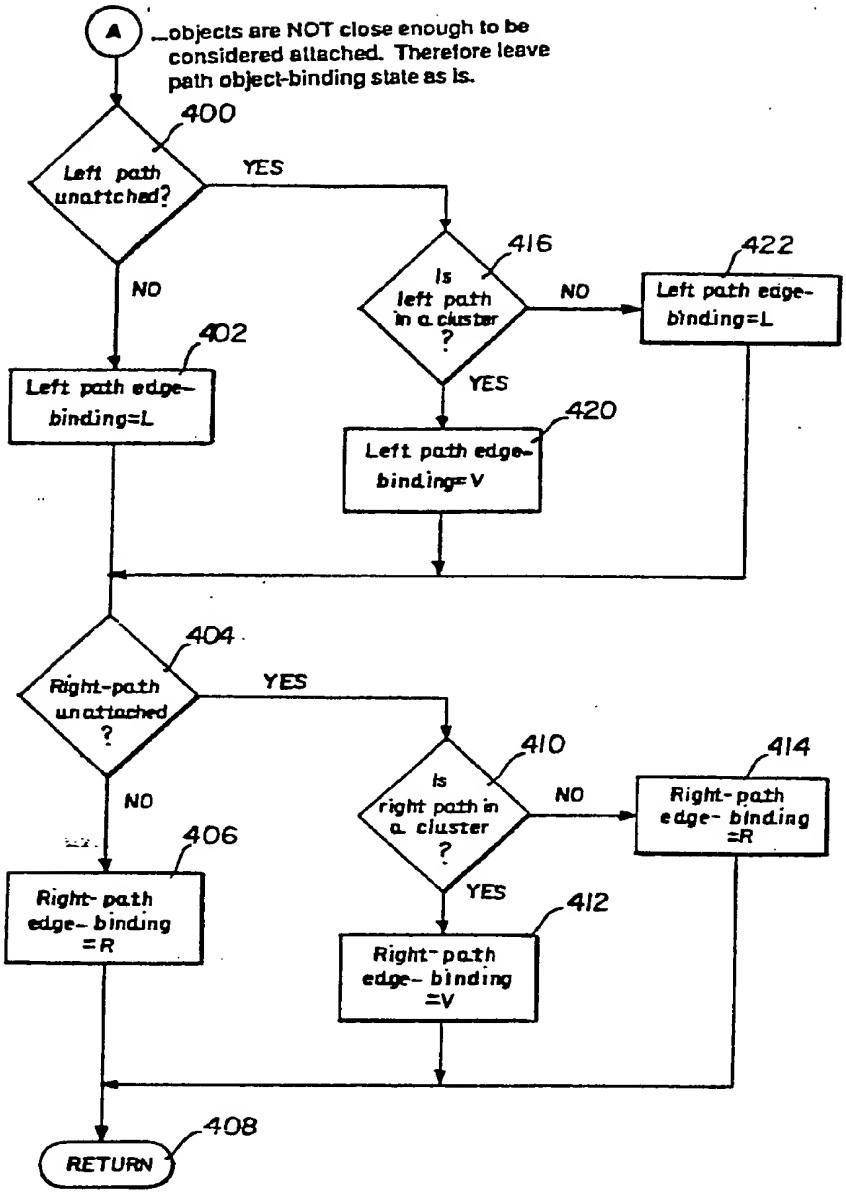
A
...objects are NOT close enough to be considered attached. Therefore leave path object-binding state as is.
400
Left path unattched?
YES
NO
416
Is left path in a cluster ?
NO
422
Left path edge-binding=L
YES
402
Left path edge-binding=L
420
Left path edge-binding=V
404
Right-path unattached ?
YES
NO
410
Is right path in a cluster ?
NO
414
Right-path edge-binding =R
YES
406
Right-path edge-binding =R
412
Right-path edge-binding =V
408
RETURN

FIG. 43